A perceptually adaptive JPEG coder

by

Henry Hoi-Yu Tong

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science,
Graduate Department of Electrical and Computer Engineering,
University of Toronto

0-612-29417-X

A perceptually adaptive JPEG coder

A thesis for the Degree of Master of Applied Science, 1997

by

Henry Hoi-Yu Tong

Department of Electrical and Computer Engineering

University of Toronto

Abstract

A perceptually adaptive JPEG coder is implemented in this research. The major part of the implementation involves the design of a perceptual model that is based on the texture masking and luminance masking properties of the human visual system (HVS). The main objective is to compute a local multiplier map that can be used to scale the quantization matrix (QM) so that fewer bits are used to represent the perceptually less important areas of an image. The texture masking model is based on a proposed block classification algorithm to differentiate between the plain, edge, and texture blocks. An adaptive luminance masking scheme is also proposed which adaptively adjusts the luminance masking strategy depending on an image's mean luminance value. Experimental results show that the adaptive coder provides savings in bit-rate over baseline JPEG, with no overall loss in perceptual quality according to a subject test.

Acknowledgments

It is my pleasure to acknowledge and thank all people who have helped me throughout this research. I would like to thank my supervisor, Prof. A. N. Venetsanopoulos for his continued support and guidance. Special thanks are due to Kostas Plataniotis for providing invaluable suggestions and ideas that have helped so much. I am also grateful to Huawu Lin, Ido Rabinovitch, Lowell Winger, and Nicos Herodotou for the extremely useful advice, suggestions, and the friendly conversations we had during the long hours in the DSP lab.

I would like to thank Prof. Katzela, Prof. Lee, and Prof. Sousa for serving in the oral examination committee. I would also like to thank Nicos and Lynval Peart for helping me prepare for the examination, and Dimitrios Androutsos and Kostas for proofreading the thesis.

Special thanks are also owed to all my friends in the communications group and the university for offering encouragement, helpful suggestions, and friendly chats. I would also like to acknowledge the Independent JPEG Group for providing the JPEG software that forms the basis source code for the research.

I wish to thank the following individuals for participating in the subjective testing and providing me with feedback and useful comments that were invaluable in completing the perceptual aspect of the thesis: Changshan Xiao, Sheldon Tam, Wing Tam, Lowell, Michael Yu, Kostas, Anestis Karasaridis, Peter Androutsos, Evan, Wai Ming Louie, Lilian Herodotou, Stanley Ma, Kevin Gior, Halim Yanikomeroglu, Chao Zeng, Yiwen, Dimitrios, Karen Cheung, and Ido.

Last but certainly not least, my special thanks to my family for their encouragement, patience and understanding, despite the many long distance phone calls that I missed during the course of this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Image compression, and more recently, video compression, have been among the main applications in communications and image processing for many years because of the enormous size of digitized images, and consequently the need to transmit or store the data in a more compact form. Despite the improvement in the capacity of high density storage devices, and the promise of high bandwidth optical transmission media, compression remains a key technology. This is because of the new demand for digital video storage, and the continued need to utilize bandlimited media such as radio and satellite links, and bit-rate-limited storage media such as CD-ROMs and solid-state memory chips [1].

The JPEG (Joint Photographic Experts Group) [2, 3] standard has been among the most important international standards for image compression. Since its standardization in the early 90s, the use of digitized image data in the computing industry has coincidentally risen dramatically. This is in large part because of the rapid advancements in computer hardware and software sophistication, but the standardization of image compression methods also assumes a pivotal role as it enables interoperability of image data among different systems, and leads to the development of cost-effective, non-proprietary implementations that will in turn further promote the use of these multimedia data [4].

Despite the popularity and success of the JPEG standard, the coding scheme utilized by the standard, a *block-based transform coding scheme* based on the *discrete cosine transform (DCT)* is still not optimal as far as compression performance is concerned. In recent years there has been increased interest in *perceptual coding*, which promises a higher compression ratio as compared to traditional coding schemes [1], among which the DCT-based transform coding schemes are the most widely used. The main difference between the traditional coding schemes and perceptual coding is that whereas the traditional coding schemes emphasize the *mean-square-error(MSE)* criterion between the original and the coded images, perceptual coding also makes use of the properties of the *human visual system (HVS)*. A main idea here is that because of the masking properties of the HVS, additional signal distortion that cannot be noticed perceptually can be introduced into certain areas of the image, thus fewer bits (and less fidelity overall) are needed to encode the image and consequently the compression ratio increases. A major weakness of the JPEG image compression scheme is that despite its good performance in the traditional MSE sense in general, it fails to take full advantage of the HVS properties to compress images more effectively. In other words, a standard JPEG coder is a rather limited perceptual coder.

1.1 Research objectives and thesis outline

The main objective of this project is the design of a perceptual model based on the *texture masking* and *luminance masking* properties of the HVS. The perceptual model is then implemented on top of a standard (non-adaptive) JPEG coder. The result is a JPEG-based perceptual coder that is able to adapt to the local statistics of different areas of an image based on the analysis from the perceptual model. The coder will also be referred to as the *A-JPEG (Adaptive-JPEG)* coder in the thesis.

The texture masking property of the HVS suggests that the human eye's sensitivity to error is low in highly textured image areas, while the luminance masking property suggests that human sensitivity to error is low in very bright and dark image regions. By using the masking properties of the HVS, the perceptual model computes a map of scalar multipliers that can be used by the perceptual coder to scale up the local quantization step sizes so that fewer bits are used to represent the perceptually less important areas of an image.

Experimental results show that with the same JPEG quality factor, the adaptive JPEG coder produces compressed images with lower bit-rates than those produced with baseline JPEG with no overall loss in perceived quality according to a subjective comparative test. One important advantage of the architecture of the adaptive coder is that the computational overhead of the perceptual model is incurred at the encoder level only. No knowledge of the perceptual model is needed for decoding or decompression. Thus the perceptual model only needs to be implemented in a JPEG encoder and the output can be read by any existing standard-compliant JPEG decoder without overhead. In addition to the obvious benefit of maintaining compatibility with the existing standard, the adaptive coder is also ideally matched for broadcast-type multimedia imaging applications where most of the information is created only once but accessed many times afterwards. In these applications, the cost of the encoding overhead of the perceptual model, in light of the overall system usage cost, will be minimal because encoding is done only relatively sparingly.

The following is the outline of the remaining chapters of the thesis. Chapter 2 reviews the basic ideas of image compression and some important human visual system properties that form the foundations of perceptual coding. Chapter 3 introduces the JPEG compression standard and the transform coding model that JPEG uses. The design of the perceptual model and the A-JPEG coder is presented in chapter 4. Chapter 5 presents the experimental results and the corresponding discussions. Chapter 6 concludes the thesis and outlines future work to be done. The appendix at the end of the thesis provides a more general discussion on how the adaptive coder might be applied in practice.

Chapter 2

Review of image compression and some human visual system properties

This chapter gives a brief review of image compression. Some relevant human visual system (HVS) properties will also be presented. The main purpose of this chapter is to establish the compression and HVS terminologies that will be referred to throughout the whole thesis.

2.1 Classical information theory

The foundation of image compression originates from Shannon's works on *Information Theory* [5], in which he defined the measure of the average uncertainty, or randomness, of a source S as the *entropy* $H(x)$:

$$H(x) = -\sum_{i} p_i log_2 p_i \,, \tag{2.1}$$

where p_i denotes the probability of occurrence of the symbol x_i from the source S. The term entropy can also be defined as the data compression limit, which corresponds to the lower limit for representation of a source. For instance, consider the representation of one pixel from an image with 8 bit pixel depth, where 256 different values (symbols) are possible for one pixel. If all values are equally likely, i.e. $p_i = \frac{1}{256}$ for all i (a totally random image with a flat distribution), the entropy $H(x)$ would be 8. The unit for entropy is bit/symbol, thus equation 2.1 shows that the lower limit for representing a pixel from this example image is 8 bit/pixel, or no compression is possible. This represents the worst case scenario for compressing an image pixel. The other extreme is, for example, $p_0 = 1$, and $p_i = 0$ for all $i \neq 0$. In this case the entropy is 0, which means that no bit needs to be transmitted because the symbol is always x_0. In reality, p_i's distribution is normally not flat and the value of p_i is somewhere between 0 and 1 for most i. From information theory, a non-flat source distribution would lead to an entropy $H(x)$ below 8, and compression is possible.

It is conventional in the image compression community to use the term ***bit-rate*** when comparing different compression schemes [6]. The term bit-rate means the average number of bits per pixel after compression. As with compression's origin from the *source coding theory* in information theory, the word *encoding* or *coding* is often interchangeable with the word compression, whereas, *decoding* naturally refers to the act of decompression.

The concept of entropy defines the lower bound bit-rate for error-free, or *lossless* compression. But in certain cases it is also useful to allow some distortion between the original and the decompressed image so that the the compressed bit-rate can be lower. This is called *lossy* compression, and the original image cannot be perfectly recovered from the compressed image. Shannon proposed the more general *rate-distortion function* which establishes the theoretical minimum bit-rate, $R(D)$, that is achievable given an average distortion D between the original and the decompressed image. The general form of a rate-distortion function [7, 5] is shown in figure 2.1. The curve shows that when D increases, $R(D)$ decreases. In other words the minimum theoretical bit-rate decreases, or higher compression is possible, when more distortion is allowed. The rate-distortion function is also applicable for lossless compression: when $D = 0$, $R(0)$ is the entropy of the source.

Classical information theory provides a solid theoretical foundation on which the

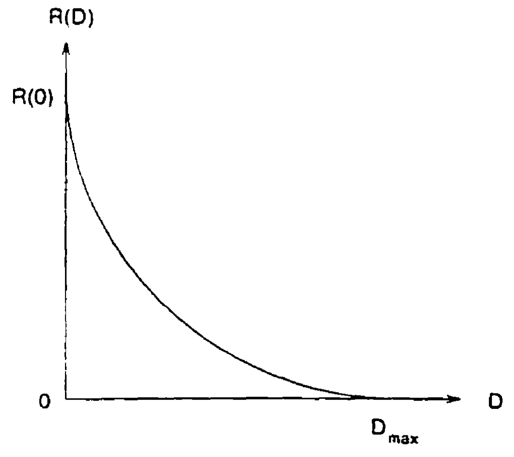


Figure 2.1: The rate-distortion function

modern compression techniques can build on. However, there are several issues concerning the use of classical information theory in practice [1, 6]:

Performance bounds only The theory is *non-constructive*, it offers bounds on distortion-rate performance rather than techniques for actually achieving these target bounds. In practice the calculation of the bounds is not simple and they are mostly used for theoretical comparisons only.

Statistical redundancy removal and input modeling The entropy measure depends entirely on a known probability distribution of the source. In the previous example, the coding of a single 8-bit pixel source uses a histogram of the counts of the 256 different possible symbols as the probability source. However, image data tend to have a high degree of spatial redundancy, i.e. adjacent pixels tend to have similar magnitudes, and the above entropy measure is sub-optimal as the inter-pixel redundancy is not taken into consideration. The calculation of the second-order entropy $H(x, y)$ requires 256 different histograms of 256 counts each, while $H(x, y, z)$ will require a staggering 65,536 different histograms. One of the main goal of compression research is to *model*, or *preprocess* the input data to take advantage of the inherent *statistical redundancy*, in particular, the *interpixel redundancy* in the image more efficiently. For example, a typical real-life image contains large areas of relatively constant grayscale values.

By coding the difference of adjacent pixel values instead of each individual pixel, the probability distribution of the differential output will be heavily skewed near zero. This distribution will very likely be more 'non-flat' than the original pixel distribution. The source variance will also be smaller and a lower coded bit-rate is achievable. This method is called *lossless differential coding* and is one of the most simple and commonly used techniques to reduce statistical redundancy.

Perceptual redundancy removal For lossy compression, the definition of the distortion measure D directly affects the $R(D)$ curve and thus the coding performance. The traditional *mean-square-error (MSE)* criterion has long been known as a poor indicator of image quality, which ultimately has to be judged by a human viewer. Research on the *masking properties* of the human visual system has shown that local image characteristics have different masking effects on the actual perceived distortion [8, 9]. In other words, the distortion is less noticeable in some area of an image than in other area - that is, there is *perceptual redundancy*, or *visual redundancy*, in an image [10]. By carefully introducing more distortion into areas that are less susceptible to error perceptually, the compressed bit-rate can be decreased without affecting the overall perceived image quality.

Figure 2.2 shows a typical classification of the well-known compression methods [4]. The *first generation compression techniques* represent the vast majority of the compression methods currently in use, including JPEG. They will be the focus in the next section.

The *second generation image compression techniques* involve using image synthesis and sophisticated image feature modeling to achieve very high compression ratio. They are designed for low-bit-rate coding, for example, video conferencing applications, where neighbouring image frames are generally very similar to each other, so only a minimal amount of feature information is needed for transmission to reconstruct a frame sequence. The decompressed image using second generation compres-

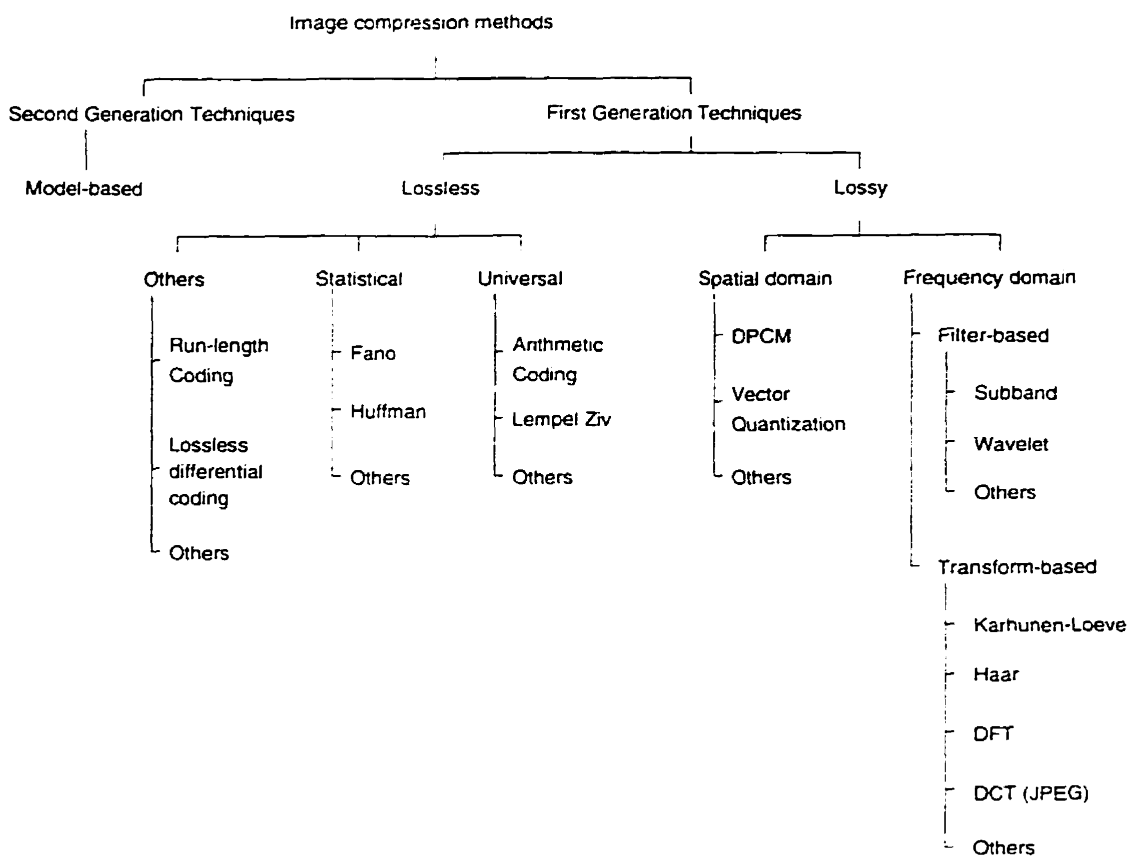


Figure 2.2: Classification of image compression methods

sion techniques is generally far from indistinguishable from the original image [7], and these techniques will not be further pursued in this thesis.

2.2 Lossless and lossy image compression

The first generation techniques can be further classified into lossless and lossy compression techniques. Lossless compression deals with statistical redundancy removal, and allows the original image to be recovered exactly from the compressed image. Lossy compression, in addition to performing statistical redundancy removal, also removes perceptual redundancy, this introduces distortion and results in imperfect reconstruction. Most lossless and lossy compression methods fit within a general framework depicted in figure 2.3 [4].

The general compression framework consists of two parts: the preprocessors and

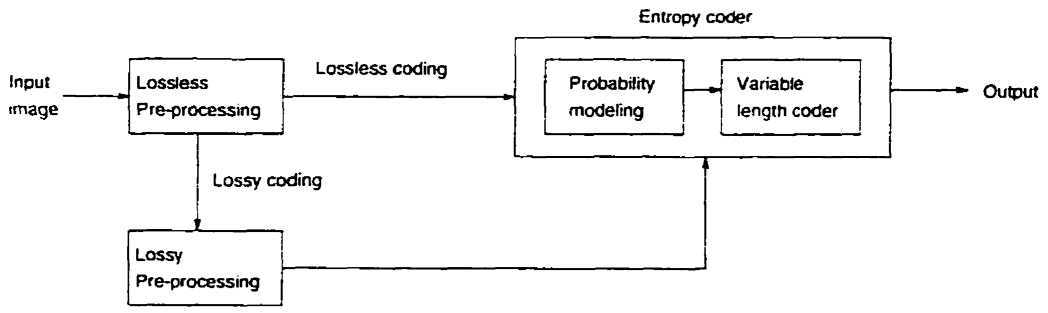


Figure 2.3: Lossless and lossy compression general framework

the entropy coder. For some compression techniques. the distinction between the different building blocks may not be obvious, nevertheless the idea behind the framework is general and should apply to most compression systems. The *entropy coder* is the simplest but yet most crucial component of most compression systems. Given an input set of symbols. the probability modeler generates a table of the probability distribution. The variable length coder then uses the table to map the input symbols into codewords. By using short codewords for symbols with high probability of occurrence and long codewords otherwise. overall compression is achieved.

The lossless entropy coder provides good performance for general data compression by itself. however. it is not very effective for complicated input like images. The main idea behind the general compression framework is to first preprocess the input image data and then entropy code the processed symbols for better performance. The preprocessing steps remove the statistical and perceptual redundancy from the input using pre-defined perceptual and statistical models, and enable very effective entropy coding.

The differential coding method mentioned previously is a typical lossless preprocessing operation. Lossy preprocessing is normally realized through *quantization*. Quantization in general refers to the operation of assigning a discrete value to represent a group of values. This *many-to-one* operation is inherently lossy. For instance, using a uniform quantizer with a quantization step size Δ, the quantizer output equals the input symbol divided by Δ. This reduces the precision of the input by $log_2\Delta$ bits, and is the main source for irreversible information loss in most lossy compression systems.

The following is a brief outline of some of the more common lossless compression techniques:

Huffman coding The Huffman coder is the first widely-used entropy coder after Shannon published his landmark papers. Its structure is identical to that of the entropy coder shown in figure 2.3. With the input symbol probabilities, the variable length symbol-to-codeword mapping is performed using Huffman's algorithm [11]. One disadvantage about Huffman coding is that the probability table also needs to be transmitted for proper decoding. Furthermore, the minimum codeword length is 1 bit, no matter how high the probability of occurrence of a symbol is. Nonetheless, combined with proper pre-modeling and quantization of the input data, Huffman coding has proven to be very effective for entropy coding and it is the entropy coder of choice for most JPEG implementations.

Runlength coding When the input contains symbol sequences of constant magnitudes, compression can be achieved easily by just coding the length of the sequence (run) and the symbol itself. For instance, ten consecutive zeros can be coded using only two symbols - a ten, and a zero. This technique is known as runlength coding. It is frequently used for simple stand-alone compression, or as a preprocessor before entropy coding is applied.

Arithmetic coding Another popular entropy coder which is an improvement over the basic Huffman coder. The coder is discussed in details in [12]. It has the advantage that the minimum codeword length can be less than 1 bit. Moreover, it allows dynamic adaptation of the probability model with the input statistics and no input-dependent probability table is needed for decoding. Arithmetic coding is also listed as an optional entropy coding method in the JPEG standard [3], however, because the algorithm is patented, it is not necessarily supported by all JPEG implementations [13].

Lempel-Ziv coding Lempel-Ziv coding and arithmetic coding are both known as *universal coding algorithms* as they are both capable of dynamic adaptation according to the input statistics and no separate probability modeling is necessary. Lempel-Ziv coding uses a *dictionary-based* approach. The idea is to adaptively build up a dictionary of the input symbol sequences and use the dictionary indexes as the codewords [14]. Lempel-Ziv coding is very popular for lossless text and image compression. For example, variations of Lempel-Ziv coding are used in the pkzip compression program and the GIF graphics format.

Some of the popular lossy compression techniques are:

Differential pulse code modulation (DPCM) DPCM belongs to the *predictive coding* family, in which a predictor is used to predict a current symbol x_t using previous symbols. The difference, or the error e_t between the current symbol and the prediction is then quantized as the output. The variance of e_t is normally lower than that of x_t, this corresponds to lower entropy, and thus higher compression. In some cases the output is subjected to entropy coding for further compression. It is a lossless differential coder if the quantization step is removed, and the output is entropy coded [14, 7].

Vector quantization Instead of quantizing individual image pixels, a vector of pixels are quantized together. In short, an image is decomposed into vectors that are matched against a codebook of possible vectors. The encoder tries to find a vector from the code book that best-fits each image vector. Like regular scalar quantization, this is a many-to-one operation through which many image vectors will be mapped into a single codebook vector index. One characteristic of vector quantization is that the decoding is very fast - the decoder just uses a look-up-table to retrieve the codebook vector from the received index. However, encoding, which generally needs to search the whole codebook for the best-fit vector, is time-consuming. Moreover, the codebook needs to be generated offline using a training set of representative images [7, 15].

Transform coding The basic operation is the transformation of a block of pixels from the spatial domain to the transform (frequency) domain. This is followed by quantization and then entropy coding. The transformation performs data decorrelation, while quantization removes perceptual redundancy. The discrete cosine transform (DCT) is the transform of choice for most transform coders. Transform coding is the compression model of the JPEG standard and will be discussed in more details in chapter 3.

Subband coding and wavelet coding Another family of frequency-domain-based approaches. Instead of orthogonal transformation, filter banks are used to split the input image into individual frequency *subbands*. One advantage of this approach over transform coding is that this is not a *block-based* approach. So the disturbing blocking artifacts common in low-bit-rate compression with transform coding is absent here. Wavelet coding can be considered as a special case of subband coding, where wavelet coding uses wavelet filters and a more specific band-splitting technique for wavelet representations of the data. Please see [16, 17] for more details.

2.3 Relevant human visual system properties

Over the years, there has been a steady increase in the extent to which knowledge about human perception has been incorporated into image compression [1]. In particular, the filtering and masking characteristics of the human visual system (HVS) are of interest. Moreover, when compressing colour images, the effects of the use of different colour spaces on the compression performance are also important. The first part of this section introduces the issues concerning colour image coding. The second part summarizes the filtering and masking properties of the HVS that form the foundations of perceptual coding.

2.3.1 Colour space and image coding

The trichromatic theory of colour vision suggests that any colour image can be represented by three colour channels [14]. Among the many colour spaces that have been designed for the representation of colour, the RGB (Red, Green. Blue) colour space has been a very popular colour space for digital image processing and general manipulation of images in the computer industry since it is the device colour space for televisions and computer monitors. In practice, even if other colour spaces are used for the processing of an image, for display purposes, the final representation of the image still needs to be in the RGB colour space as a RGB signal can be displayed directly on a monitor without other colour transformation [10, 18].

One disadvantage of using the RGB colour space for image coding is that the energy contents in the three colour channels are approximately the same. Thus each channel carries about the same weight and the potential correlation between the three channels cannot be exploited. A common alternative to the RGB colour space is a luminance-chrominance colour space such as the YIQ colour space. *Luminance* stands for the brightness of an image, while *chrominance* stands for colour information and typically consists of two colour channels. In the YIQ colour space, Y is the luminance channel, and I, Q are the chrominance channels.

The YIQ model was designed to take advantage of the HVS's greater sensitivity to changes in luminance than to changes in chrominance in colour images [10]. The colour space transformation decorrelates the input RGB channels so that the chrominance channels have lower energies and lower bandwidths as compared to the luminance channel Y [19]. This permits the use of subsampling as well as the use of coarser quantization for the chrominance channels and results in much improved compression ratios for colour images [1, 18]. In practice, the overall proportion of the chrominance information can be as low as only 10% of the total bit-rate with negligible loss in the perceived image quality [1] and this is also an example of the application of perceptual redundancy removal in an image coding system.

In JPEG image compression, another variation of the luminance-chrominance colour space, the YCrCb colour space is normally used [3]. In this case, Cr, Cb stand for the red and blue chrominance information respectively. The matrix for the linear transformation from RGB to YCrCb is:

$$\begin{bmatrix} Y \\ Cr \\ Cb \end{bmatrix} = \begin{bmatrix} 0.2989 & 0.5866 & 0.1145 \\ 0.5000 & -0.4183 & -0.0816 \\ -0.1687 & -0.3312 & 0.5000 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} . \tag{2.2}$$

and the reverse process is a similar linear transformation.

The YIQ and YCrCb luminance-chrominance colour space systems are also used in commercial colour television broadcasting. In addition to the aforementioned advantage of bandwidth, or bit-rate conservation, they were also developed for maintaining compatibility with monochrome black-and-white TV standards [10]. The luminance channel Y provides all the information required by a monochrome TV set. In fact, a monochrome grayscale image is just a single channel image that contains the luminance channel of its colour counterpart.

Most the past human visual system research concerning perceptual coding has been concentrated on the response of grayscale images [14, 19]. One main reason is that, as explained above, the luminance channel contains the majority of the image energy and the information that can be retrieved from the chrominance channels is relatively limited. Moreover, multi-channel colour vision is still not well understood as compared to the single-channel case for grayscale images [14]. There is also hardware constraint as full-colour imaging device has not been widely available until relatively recently. As a result the filtering and masking properties that will be discussed in the following section are the luminance properties of the human visual system.

2.3.2 The filtering and masking properties

This section summarizes the relevant filtering and masking properties of the HVS briefly. Of the three HVS properties in discussion, *frequency sensitivity* is a filtering characteristic, whereas the others are masking properties.

The *visibility threshold* is an important measure for quantifying perceptual redundancy. It can be defined as the magnitude of a stimulus in an image at which it becomes just visible or just invisible [14, 1, 20]. The stimulus can be a sinusoidal function or an arbitrary form of additive noise or distortion. In theory, any stimulus that is below the local visibility threshold can be eliminated (or in the case of distortion, tolerated), which results in perceptual redundancy removal. In some circumstances, instead of the visibility threshold, the measure of the *sensitivity* of the human eye to certain stimuli may be needed. Sensitivity is simply the inverse of the visibility threshold, i.e. the higher the visibility threshold, the lower the sensitivity. In some other cases, the visibility of a stimulus may be described. Similar to the sensitivity measure, visibility is also inversely related to the visibility threshold.

Frequency sensitivity Psychovisual studies have shown that the perception of distortion depends on the human visual system's frequency response. Experiments have been conducted to measure the visibility of sinusoidal functions of varying magnitudes. It was shown that the human eye acts as a bandpass filter, with a maximum response (or sensitivity, in the image coding context) in the range of two to eight cpd (cycles per degree), falling off at lower, and especially higher frequencies [8, 21]. The response curve is called the *modulation transfer function (MTF)*, of which the general shape is shown in figure 2.4. Frequency sensitivity

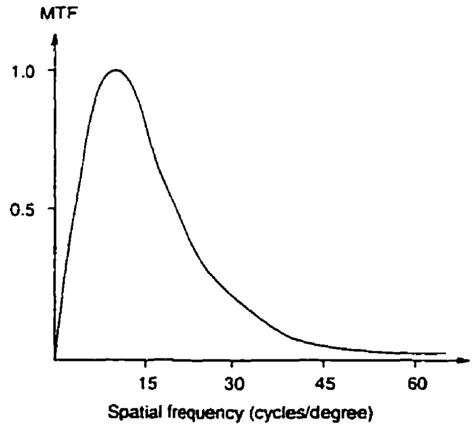


Figure 2.4: The modulation transfer function

provides a natural way to incorporate perceptual criteria into transform coding, in which the HVS response can be used to weight the relative importance of the transform coefficients. The higher the weight, the more important a coefficient is. Conversely, a coefficient with a low weight can be quantized coarsely. or simply discarded [22].

Luminance masking In this thesis. the terms *luminance* and *brightness* will be used interchangeably. This will suffice for the current work, even though straightly speaking the luminance value is a physical measure, while the term brightness is a subjective descriptor that cannot be measured [10, 23]. In addition. the term *grayscale*, which refers to the luminance component of a digital image. will also be used to represent brightness or luminance. For instance, an 8-bit grayscale value of zero means total darkness, or the lowest luminance; while the maximum 8-bit grayscale value of 255 means bright white, or the highest luminance. The concept of luminance masking is illustrated in figure 2.5 [14]. The dashed area is perturbed by the magnitude ΔL at which the perturbation

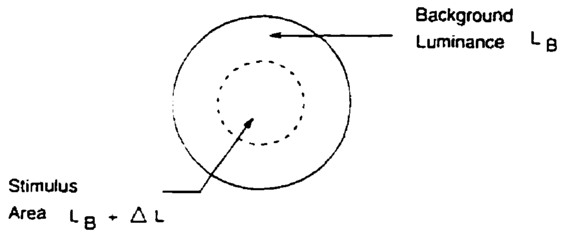


Figure 2.5: Background luminance and the visibility threshold

is just visible. Experiments show that the visibility threshold ΔL is a function of the background luminance L_B and it increases almost linearly with L_B. This is known as *Weber's law* [14]:

$$\frac{\Delta L}{L_B} = constant\ (the\ Weber\ fraction) \tag{2.3}$$

The implication from Weber's law is that the human eye is less sensitive to errors in the bright areas (areas with high luminance values) of a picture because ΔL is relatively high in those areas. Weber's law is generally accurate over the normal range of middle-low to high luminance values. However, in very dark area, it has been reported that the Weber fraction tends to increase with decreasing background luminance values [1, 14, 21]. In other words, the human eye's sensitivity to distortion also decreases in very dark area. Figure 2.6 shows a graph of the distortion visibility vs. the background brightness [1].

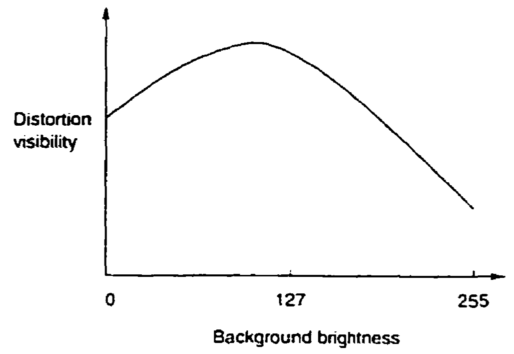


Figure 2.6: Distortion visibility vs background brightness

In practice, luminance masking is very useful for image coding. As the visibility threshold is relatively high in very bright and dark areas, perceptual redundancy exists in those area of an image, and more local distortion can be introduced for higher compression.

Texture masking This is also called *spatial masking*. In this case the visibility of distortion decreases when there is a large visible change in the luminance background. Figure 2.7 shows an example of the masking effects [14].

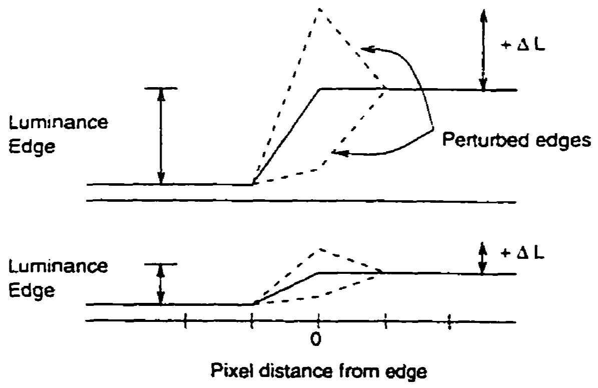


Figure 2.7: Texture (spatial) masking effects

A large luminance edge represents a rapidly changing background. The visibility threshold ΔL of the perturbed edge is larger with a higher edge. Netravali [14, 9] shows that the presence of high contrast edges can increase the visibility threshold considerably. In perceptual image coding, texture masking has been used extensively with spatial domain based coding system such as DPCM, in which the high luminance edges can be located exactly and thus quantized more coarsely. Texture masking can also be used in block-based transform coding system, where the texture energy can be approximated by the L_1 or L_2 norm of the transform coefficients of the local block. High texture energy implies a block with rapidly changing luminance values in the spatial domain, thus perceptual redundancy exists and additional compression is possible in the local block.

In addition to the three properties discussed above, *temporal masking* is another well-known masking property of the HVS [1, 24]. It indicates that during video playback, when there is scene change or object movement, the distortion visibility is low in the newly displayed areas for a short latency period and therefore perceptual redundancy can be exploited. However this property concerns with the time domain property of video and can be used in video coding only. The focus of this project is still image coding and thus temporal masking will not be further pursued in this thesis.

Chapter 3

The JPEG image compression standard

The JPEG standard is designed for compressing continuous-tone photographic images, both grayscale and colour. The standard provides four operating modes. This chapter describes the key concepts of the most common *baseline sequential* mode. The baseline JPEG mode specifies the minimum set of requirements for a JPEG compliant implementation and it covers all the important features of the standard. For more detailed discussion, please refer to [2, 3]. The block diagram of a baseline JPEG compression system is shown in figure 3.1.

The JPEG standard requires that the input image be separated into blocks of 8 by 8 pixels and JPEG compression can be regarded as the compression of a stream of 8x8 grayscale image blocks. The standard also supports multiple channel image compression, and colour image compression can be regarded as the compression of three grayscale images, which are normally compressed by alternately interleaving groups of 8x8 blocks from each channel [2].

The JPEG standard is not colour-space specific, thus it can be used for compression of images in any colour space. In practice, most JPEG implementations compress colour images using a luminance-chrominance colour space to take advantage of the narrow-bandwidth characteristic of the chrominance channels as discussed in section 2.3.1. The original RGB colour space is converted to the YCrCb colour

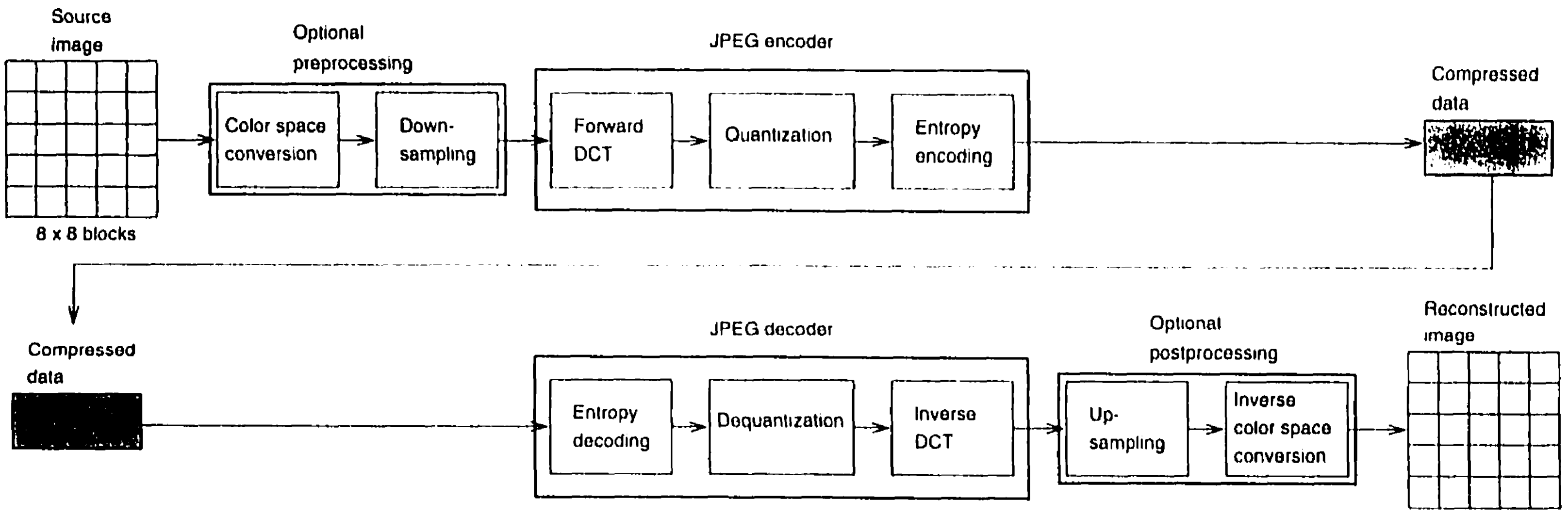


Figure 3.1: A baseline JPEG compression system

space using equation 2.2. and the Cr, Cb chrominance channels are subsampled two to one both horizontally and vertically before JPEG compression is performed. The operations on colour space and sampling rate conversions are not part of the JPEG standard requirements and are shown in the preprocessing and postprocessing boxes in figure 3.1.

3.1 The transform coding model

The core JPEG compression operations: discrete cosine transformation, quantization, and entropy coding form the basic components of the transform coding model and are introduced in this section.

3.1.1 Transformation

In transform coding, the statistical redundancy in a block can be modeled as the interpixel correlation, and the main goal is to perform *decorrelation* on the block by a transformation operation. JPEG uses the *Discrete Cosine Transform (DCT)*, which closely approximates the decorrelation performance of the optimal transform (the *Karhunen-Loève Transform (KLT)*) for image blocks that are *smooth* [6]. This smoothness property is also a general characteristic of the target image group of the JPEG standard: continuous-tone photographic images; in fact DCT has enjoyed enormous popularity as the transformation of choice for most transform-based coding systems.

The following are the equations of the 8x8 two-dimensional forward DCT of the image block $f(x,y)$, and the inverse DCT of the transform coefficient block $F(u,v)$:

$$\begin{aligned} F(u,v) &= \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y) cos\frac{(2x+1)u\pi}{16} cos\frac{(2y+1)v\pi}{16} && (3.1) \\ f(x,y) &= \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)F(u,v) cos\frac{(2x+1)u\pi}{16} cos\frac{(2y+1)v\pi}{16}, && (3.2) \end{aligned}$$

where

$x, y, u, v = 0, \ldots, 7;$

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

The DCT coefficient block $F(u, v)$ is a function of the horizontal and vertical *spatial frequencies* u and v respectively. The value $F(0, 0)$, at the lowest frequency $(0, 0)$, is called the *DC coefficient*, and is equivalent to the mean value of the 64 pixels. The remaining 63 $F(u, v)$'s are called the *AC coefficients*[1].

The DCT is a reversible transform apart from roundoff errors, which are generally negligible. It also has a nice *energy packing* property: it is almost as good as KLT in packing the most energy in the fewest possible number of transform coefficients. In practice, for the majority of image blocks, after transformation most of the block energy is packed in a few low frequency coefficients only. This is a crucial preprocessing step that leads to more efficient perceptual and statistical redundancy removal in the later coding procedures:

- The high frequency DCT coefficients are relatively insignificant, so they can be more coarsely quantized with little effects on the overall perceptual quality.
- After quantization, most of the high frequency coefficients are zeros or near zeros. The last step, entropy coding, can take advantage of the new redundancy of zeros and compress very effectively.

3.1.2 Quantization

Each of the 64 DCT coefficients $F(u, v)$ is uniformly quantized by a corresponding entry $Q(u, v)$ from an 8x8 quantization matrix:

$$F_Q(u, v) = Round\left(\frac{F(u, v)}{Q(u, v)}\right), \tag{3.3}$$

[1]Strictly speaking the frequency notation is only correct for the output of the discrete Fourier transform (DFT). Nevertheless DCT is closely related to DFT and it is intuitive to regard the DCT coefficients as the relative weights of 64 2-D DCT basis vectors arranged in the order of rapidity of change, which corresponds loosely to the notion of frequency (see p.59 in [7]). For instance, $F(7, 7)$ represents the weight of the DCT basis vector that has the most amount of activity, or the highest frequency.

and the dequantization operation for reconstructing the DCT coefficients at the decoder is:

$$F'(u,v) = F_Q(u,v) \times Q(u,v). \tag{3.4}$$

Each quantization matrix entry $Q(u,v)$ is an integer from 1 to 255 which specifies the quantization step size for the DCT coefficient at frequency (u,v). The maximum quantization error for reconstructing $F(u,v)$ is then $\frac{Q(u,v)}{2}$. The quantization stage is the only lossy operation in JPEG and it is responsible for removing perceptual redundancy. The goal is to reduce the precision of the coefficients that are visually insignificant.

The quantization matrix (QM) is the most crucial component in the quantization stage. There are basically two general ways of designing the QM: One through the use of the rate-distortion theory, the other is based on psychovisual experiments and properties of the human visual system [4]. The first general technique calculates an image-specific QM by allocating bits to each DCT coefficient using rate-distortion criteria, given a total bit budget. Typically the low frequency coefficients are allocated many more bits because of the energy compaction property of DCT. Examples of this technique can be found in [25, 26]. One problem with this technique is that the QM generated is image-dependent and the encoding overhead for computing the QM needs to be taken into consideration.

The second technique is based on human perception. The idea is to determine the visibility threshold for each DCT coefficient so that optimally any quantization distortion is not discernible. Typically the quantization step sizes for the high frequency coefficients are much larger than those for the low frequency coefficients. This leads to higher distortion for the high frequency coefficients, and agrees with the frequency sensitivity characteristic of the HVS. The JPEG standard provides a set of example quantization matrices that was designed using the psychovisual experiment described in [27]. The standard does not mandate the use of any specific QM, nevertheless the example QMs have been used as the de facto QMs in a lot of JPEG implementations and experience has shown that they are quite robust and are applicable to a wide range of applications[4].

JPEG only allows one global quantization matrix per colour channel. in other words. the QM in use cannot be changed within one channel. The implication of this for doing adaptive coding with JPEG will be discussed in chapter 4. Figures 3.2 and 3.3 present the JPEG example QMs for compressing the luminance and chrominance channels respectively. The luminance QM can also be used for the compression of single-channel grayscale images.

Scaling the compression performance

A JPEG user can control the output bit-rate by supplying different QMs. which are also included with the compressed data for decoding. The higher the quantization step sizes in the QMs. the higher the compression ratio and distortion. Most JPEG implementations provide a *quality factor* with which the user can control the bit-rate. Please note that the JPEG standard does not contain a 'quality factor' parameter. The quality factor's only purpose is to provide a convenient interface for scaling up or down the QMs during encoding for a desired bit-rate and image quality. The decoder only needs the scaled QMs to reconstruct the image.

The quality factor of a JPEG encoder is basically a proprietary measure and its meaning varies across different implementations. e.g. a quality factor of 1 might mean the best quality in one implementation. but the worst quality in another case. This project uses the JPEG software provided by the *Independent JPEG Group (IJG)* [13]. which is a very popular JPEG implementation. The IJG JPEG encoder uses a quality factor in the range of 1 to 100. with 100 for best quality. The quality factor is used to generate a *multiplying factor* which is used to linearly scale the QMs. by default the JPEG example QMs. For example. a multiplying factor of value 0.5 divides all the entries in the QMs by 2. The algorithm for converting the quality factor to a multiplying factor is as follows:

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
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

Figure 3.2: The JPEG luminance quantization matrix

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	66	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	99	99	99	99	99	99	99

Figure 3.3: The JPEG chrominance quantization matrix

```
if $(QualityFactor \geq 50)$
        $temp = 200 - QualityFactor \times 2;$
else
        $temp = 5000 \div QualityFactor;$

$MultiplyingFactor = temp \div 100;$
```

Figure 3.4 shows the graph of the QM multiplying factor as a function of the JPEG quality factor (Q). At the best quality when Q = 100, all entries in the quantizatoin matrix are ones, which means that no quantization is performed.

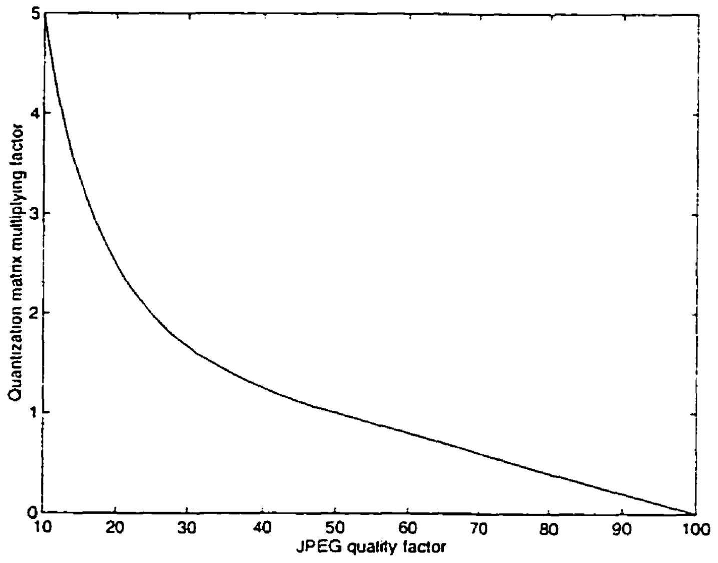


Figure 3.4: The QM multiplying factor as a function of the JPEG quality factor

3.1.3 Entropy coding

Preprocessing of the DC and AC coefficients

After quantization, the DC and AC coefficients are preprocessed differently before they are entropy coded. The DC coefficient is differentially coded by coding the difference between the current and the previous DC coefficients. This takes advantage of the remaining correlation between adjacent blocks as the DC coefficient represents the mean value of a block. Typically the variance of the differential output is much lower than that of the original DC values.

The AC coefficients are ordered into a *zig-zag* sequence as shown in figure 3.5. This

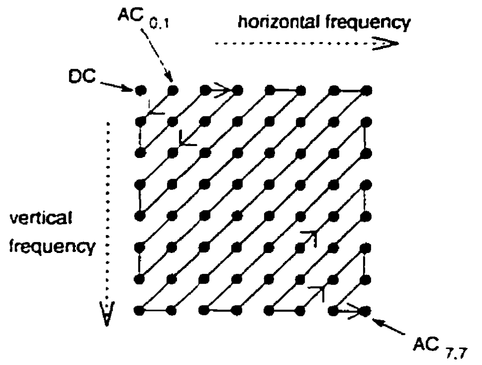


Figure 3.5: The zig-zag ordering of AC coefficients

arranges the AC coefficients from low frequency to high frequency. The sequence is then *zero-runlength coded* by representing it with the runlengths of zeros, and the non-zero coefficients. The zig-zag sequence facilitates very effective runlength coding by grouping together high frequency coefficients, which are more likely to be zeros [2].

Variable length coding

The preprocessed data: the differential DC coefficients, and the zero-runlength representations of the AC coefficients, are then Huffman coded using a DC Huffman table and an AC Huffman table respectively. For details please see [2, 3, 10]. The JPEG standard also supports using arithmetic coding for entropy coding. However the baseline JPEG mode only supports Huffman coding; moreover, in practice few implementations support arithmetic coding because of patent and subsequently license fee concerns [4, 13].

The Huffman tables need to be sent to the decoder for proper decoding. As in the case for quantization matrices, the standard lists a set of example Huffman tables for reference purposes, and it does not specify any default set. But unlike the quantization matrix, a customized, image-dependent Huffman table is rather straightforward to generate - an additional statistics-gathering pass is needed prior to entropy coding. The IJG JPEG software uses the JPEG example Huffman tables as default, but also provides an option to generate customized Huffman tables.

Chapter 4

Design of the adaptive JPEG coder

This chapter presents the design of the perceptually adaptive JPEG (A-JPEG) coder. The first section introduces the coder and its two adaptive coding modes. It is followed by a detailed discussion of the design and implementation of the perceptual model and other coder components.

4.1 Perceptually adaptive JPEG coding

As discussed in chapter 3, JPEG's quantization matrices take into account the frequency sensitivity of the HVS by more coarsely quantizing the high frequency DCT coefficients. However, the other two masking properties discussed in chapter 2: luminance masking, and texture masking are not exploited by the JPEG standard at all, and this project's primary goal is to incorporate these two local masking properties into the JPEG compression model for improved compression. The perceptual model of the coder operates on the luminance channel of the colour images, as the transformation to the luminance-chrominance colour space has already concentrated most of the signal energy in the luminance channel Y as explained in section 2.3.1.

It is insightful to divide the HVS properties into those that are dependent on local image characteristics, and those that are not. In Safranek and Johnston's well-known perceptual model, they are also called the *local* and *global* properties respectively [28, 29]. Frequency sensitivity, which is based on the measure of the modulation tranfer function of the HVS. can be considered a global property. Luminance masking is a local property that depends on the local average background brightness; and texture masking is also a local property that depends on local image activities.

Safranek defines a general perceptual model using the expression [29]:

$$Masking(u, v, k) = Global(u, v) \times Local(u, v, k), \tag{4.1}$$

where $Masking(u, v, k)$ corresponds to the masking level at frequency (u, v) for the kth block. The higher the masking level, the more tolerant the HVS is to distortion at (u, v). $Global(u, v)$ represents the base masking level, and $Local(u, v, k)$ represents the local multiplicative elevation factors that affect the masking levels. The idea is that the masking levels in some areas of an image can be elevated because of the HVS's local masking properties. The goal is to decrease the overall compressed bit-rate by adaptively introducing more distortions into image areas that are less susceptible to distortions perceptually.

The perceptual model for this project is a loose variation of the above general model. The masking levels can be considered as the allowable local quantization step sizes for each individual blocks. The global masking levels are represented by the entries in the global JPEG quantization matrix $Q(u, v)$, and the locally adaptive QM is:

$$Q_p(u, v, k) = \begin{cases} Q(u, v) \times m(k) & \text{if } u, v \neq 0 \\ Q(0, 0) & \text{else,} \end{cases} \tag{4.2}$$

where $Q_p(u, v, k)$ represents the local quantization matrix for the kth block, and $m(k)$ is the local multiplying factor. The problem then is to find $m(k)$ which elevates $Q(u, v)$ to exploit the local masking properties. Note that the multiplying factor is uniform across all frequencies in the block except for the DC coefficient at frequency $(0, 0)$, where $m(k)$ is always 1. This avoids excessive quantization of the important DC coefficient, which still maintains a certain level of correlation with adjacent blocks.

This perceptual model is general in nature and thus may be used in any DCT-based block transform coding schemes to provide perceptual information about a target image. In particular, the structure of the perceptual model is designed so that by using only a local multiplying factor for local adaptation in each block, compatibility with the JPEG standard and its extension [30] can be easily achieved, as will be described in the next section.

4.1.1 The two adaptive coding modes

As mentioned in the previous chapter, JPEG only allows one global QM per colour channel and there is no direct facility in the original standard for adaptively changing the quantization matrix values. Nevertheless since JPEG's standardization in the early 90s, there has been active interest in adaptive JPEG coding in the literatures and the JPEG standard extension also specifies added parameters for adaptive scaling of the global quantization matrix [30]. Thus there are in general two different ways to perform adaptive coding in a JPEG-style coding system: one involves the inclusion of the local multiplier map as overhead information with the compressed bit stream, and the other that uses the local multipliers to threshold the DCT coefficients without the need to send the overhead information. The A-JPEG coder implements both modes for adaptive coding and the two modes will be discussed in the following sub-sections.

4.1.1.1 The JPEG compatibility mode

A common way (in fact, the only way, to the best of the author's knowledge) to perform adaptive JPEG encoding while maintaining baseline JPEG decoder compatibility is to selectively *zero-out* some insignificant DCT coefficients just before the quantization step [1, 3, 31, 29].

The process is referred to as *adaptive thresholding*, where selected DCT coefficients are thresholded to zero using the entries of the adaptively scaled quantization matrix $Q_p(u, v, k)$:

$$F_{AT}(u,v) = \begin{cases} F(u,v) & \text{if Round}\left(\frac{F(u,v)}{Q_p(u,v,k)}\right) \neq 0. \\ 0 & \text{else.} \end{cases} \tag{4.3}$$

The block diagram for the baseline JPEG-compatible adaptive JPEG coder is shown in figure 4.1. The diagram can simply replace the original JPEG encoder block in figure 3.1. The perceptual model and the adaptive thresholding blocks together can

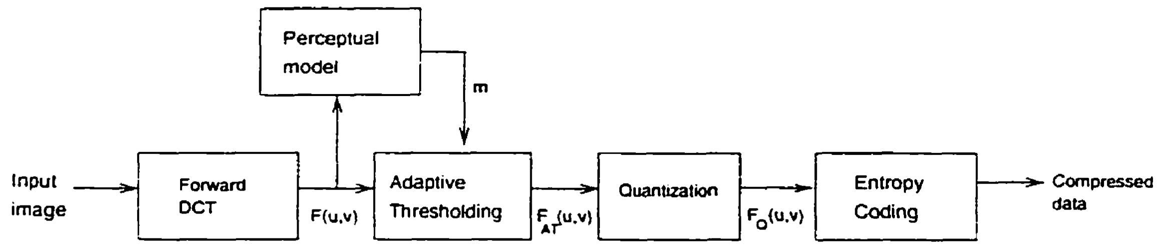


Figure 4.1: The baseline JPEG-compatible encoder

be thought of as a HVS-based preprocessing step before quantization. The perceptual model block computes the local multiplying factor using the pre-quantized original DCT coefficients. When the factor is greater than one, the quantization step size entries of the scaled QM will be greater than those of the global QM and typically compared with non-adaptive quantization, a few more zeros will be introduced by adaptive thresholding, which results in lower bit-rate overall.

The perceptual model only needs to be implemented in the encoder level. The global quantization matrix is still used exclusively in the quantization step during encoding. Thus no overhead information is needed in the dequantization step during decoding, and any JPEG-compliant decoder can be used.

4.1.1.2 The overhead mode

One limitation of doing adaptive coding in the JPEG compatibility mode is that it is a compromised approach that only affects DCT coefficients with small amplitudes by thresholding them to zeros. Larger DCT coefficients that do not get thresholded to zero are unaffected by adaptive thresholding.

Thus to examine the full effects of adaptive coding, another adaptive coding mode is implemented in which the local multipliers are used to scale the QM directly during quantization. This is referred to as *adaptive quantization* in this thesis and the block diagram for this mode of the adaptive JPEG coder is shown in figure 4.2. Another main difference between this adaptive coding mode and the JPEG compatibility mode

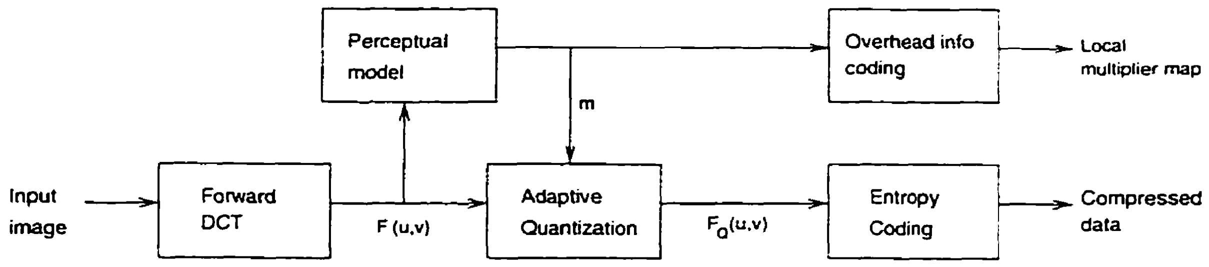


Figure 4.2: The overhead mode encoder

introduced in the previous section is that the local multiplier information also needs to be sent to the decoder as overhead information for proper decoding. Thus this mode will be referred to as the *overhead mode* from now on in the thesis.

In 1994. the draft version of the JPEG standard extension was released [30]. It introduces a *variable quantization extension* that specifies a 5-bit scalar multiplier that can be used to locally scale the global QM during quantization. The decoder can then use the multipliers to perform dequantization and reconstruct the image. Thus the quantization operation of the overhead mode of the A-JPEG coder is equivalent to the variable quantization specified in the JPEG extension and the performance of A-JPEG under the overhead mode will also provide insight into the performance of the new JPEG extension as well.[1] Additionally. the adaptive quantization operation performed in the overhead mode is also very similar to the adaptive quantization schemes of other popular DCT-based video coding schemes. thus the performance of the overhead mode of A-JPEG may also provide insight for the design of those schemes too. [32, 33, 34]

From figure 4.2, it can be observed that the adaptive quantization block replaces the original quantization block in the baseline JPEG coder. The DCT coefficients $F(u,v)$'s are quantized to:

$$F_Q(u,v) = Round\left(\frac{F(u,v)}{Q_p(u,v,k)}\right). \tag{4.4}$$

[1]Both of the JPEG extension and the overhead mode of the A-JPEG coder are not compatible with baseline JPEG since new extended decoder capabilities are needed during decoding for proper interpretation of the multiplier overhead. In spite of the added flexibility the JPEG extension provides for adaptive coding, the author has not been aware of much industry support for the new extention. possibly due to the compatibility complexity it incurs.

where the global quantization matrix is replaced by the perceptually scaled QM $Q_p(u, v, k)$ defined in equation 4.2. The decoder needs both the global quantization matrix and the multiplier map as overhead information to reconstruct the image. Furthermore, the overhead information needs to be taken into consideration when considering the overall compressed bit-rate. Note that despite the use of the multipliers for decoding, the decoder does not require any further knowledge of the internal operation of the perceptual model. Thus the decoder is also free of the computational overhead of the perceptual model as in the case for the JPEG compatibility mode.

4.1.1.3 The difference between the two adaptive modes

Figure 4.3 illustrates the difference between the JPEG compatibility mode and the overhead mode in terms of quantization behaviour, using the same perceptual model parameters. In the figure, blocks (a), (b), (c), and (d) represent an example block with the original DCT coefficients, the quantized coefficients after baseline JPEG quantization with the global QM, the scenario after adaptive thresholding in the JPEG compatibility mode, and the scenario after full adaptive quantization in the overhead mode respectively.

It can be observed that compared to baseline JPEG quantization (block (b)), adaptive thresholding (block (c)) introduces a few more zeros among the medium frequency DCT coefficients. However the non-zero coefficients are still the same as their counterparts in block (b) and they can be de-quantized using the global QM. A full adaptive quantization as shown in block (d) produces the zero AC coefficients as in block (c), in addition the magnitudes of the non-zero AC coefficients after quantization are smaller overall because a scaled-up quantization matrix is used. The smaller magnitudes provide additional compression since the Huffman codes for the small quantized coefficients are normally shorter than those for the larger ones. However the trade-off is that the scaling factor used for scaling up the quantization matrix also needs to be sent to the decoder for proper reconstruction.

-346	-179	-79	117	23	12	1	8
90	93	-225	43	80	-25	9	-13
17	71	6	-86	90	13	-21	1
-38	22	10	-49	-7	33	-13	-12
13	-3	12	-1	-17	6	6	-1
-10	-1	1	-6	-9	-2	0	12
7	4	3	-9	0	-5	4	8
1	3	1	-2	-8	1	0	2

(a) Original DCT block

-38	-30	-13	13	2	0	0	0
13	13	-28	4	5	0	0	0
2	10	0	-7	4	0	0	0
-5	2	0	-3	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

(c) After adaptive thresholding (JPEG compatibility mode)

-38	-30	-13	13	2	1	0	0
13	13	-28	4	5	-1	0	0
2	10	1	-7	4	0	-1	0
-5	2	1	-3	0	1	0	0
1	0	1	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

(b) After baseline JPEG quantization

-38	-13	-6	6	1	0	0	0
6	6	-13	2	2	0	0	0
1	4	0	-3	2	0	0	0
-2	1	0	-1	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

(d) After full adaptive quantization (Overhead mode)

Figure 4.3: JPEG compatibility mode vs overhead mode

4.2 Design of the perceptual model

The global quantization matrix $Q(u, v)$, and the local multiplying factor $m(k)$, which defines the local quantization matrix defined in equation 4.2 and reproduced as follows:

$$Q_p(u, v, k) = \begin{cases} Q(u, v) \times m(k) & \text{if } u, v \neq 0 \\ Q(0, 0) & \text{else,} \end{cases}$$

completely defines the quantization behaviour of the perceptually adaptive JPEG coder. This project concentrates on the design of the local multiplying factor $m(k)$ using the texture masking and luminance masking properties of the HVS. The example quantization matrices from the JPEG standard are used as the global quantization matrices in this project.

The perceptual model represents the local multiplying factor $m(k)$ for the kth block with the equation:

$$m(k) = TextureMask(k) \times LuminanceMask(k), \tag{4.5}$$

and the problem is to find the two local elevation factors $TextureMask(k)$ and $LuminanceMask(k)$ using the pre-quantized DCT coefficients $F(u, v)$'s. Section 4.2.2 and 4.2.3 present the texture masking model and the luminance masking model respectively for computing the two factors.

For colour images, the perceptual model computes the $m(k)$ map using the local statistics from the luminance DCT coefficients since the luminance channel contains most of the image energy. The same $m(k)$ map is then used for adaptive coding for the two chrominance channels. Since the chrominance pixels are subsampled two to one both horizontally and vertically, four luminance blocks correspond to one chrominance block in each chrominance channel and section 4.2.4 presents the algorithm for adapting the luminance $m(k)$ map for use in the chrominance channels.

4.2.1 Linear threshold elevation modeling

A simple linear threshold elevation model[35] is used as a main tool for the calculation of the local elevation factors. Figure 4.4 shows the model depicted with a curve. The

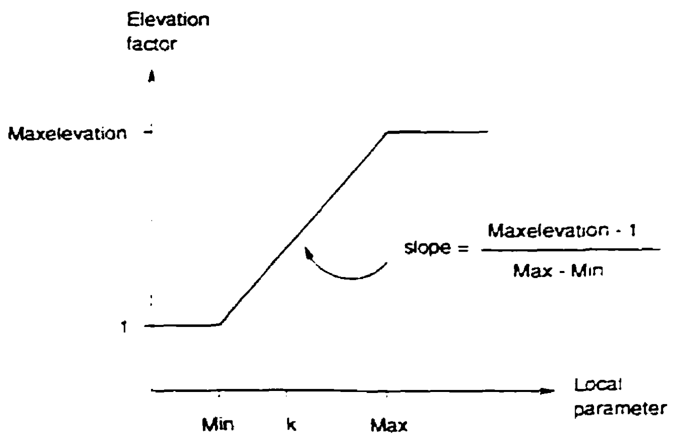


Figure 4.4: Linear threshold elevation modeling

model calculates the local elevation factor as a function of a varying local parameter in the range of the parameters Min to Max. For instance, for a local parameter k, the local elevation factor m is:

$$m = \frac{Maxelevation - 1}{Max - Min} \times (k - Min) + 1. \tag{4.6}$$

The model's elevation behaviour can be modified by the user easily by adjusting the $Maxelevation$ parameter. This provides a very convenient method to calibrate the perceptual model. The next two sections in this chapter present the details of using the linear threshold elevation model for calculations of the texture masking and luminance masking elevation factors.

4.2.2 The texture masking model

As shown in chapter 2, the human eye is less sensitive to distortions in blocks with high complexity or texture activities. This allows the adaptive coder to scale up the quantization matrix in complex blocks for higher compression ratio. According to the local image statistics, the texture masking model computes the local $TextureMask(k)$

factor that is used in equation 4.5.

The AC energy of the DCT coefficients has long been used as an indicator of the local block activity in DCT-based transform coding schemes [36]. It provides a convenient measure of block activities as the DCT coefficients are readily available as input to the perceptual model. In practice, however, not all high energy blocks are texture blocks that can be quantized coarsely. An image block with a strong edge typically has high energy too, but the distortions in an edge block will be more visible than that in a randomly textured block.

Hence in addition to AC energy calculation, a more robust model is needed to more accurately characterize the statistics of each block. A *classified quantization* scheme is thus proposed to address the issue. The goal is to *classify* each block into one of several *statistical activity classes* [37] so that appropriate quantization strategies can be designed for each class separately.

The proposed scheme can be divided into two simple steps:

1. *Block classification* is performed to classify a block into one of three classes: Plain, Edge, and Texture.

2. The $TextureMask(k)$ parameter is then computed accordingly using the classification result.

4.2.2.1 Block classification

From a historical perspective in image coding, block classification has been used extensively in two main areas. In classified vector quantization, different codebooks are designed for different block classes so that the size of each codebook is smaller than that of the otherwise general codebook for the whole image, and the codebook search time during encoding can be reduced [38, 39, 40, 41]. Block classification has also been frequently used in adaptive quantization schemes for transform-coding-based block coders [36, 42, 43, 44, 45, 46].

The block classification scheme in this project is a modified version of the scheme proposed in [43]. First the DCT coefficients of the target block are divided into four areas as shown in figure 4.5. The absolute sums of the DCT coefficients in the

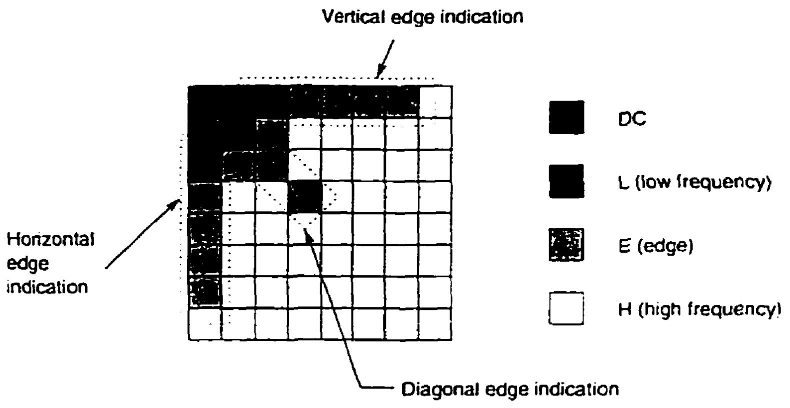


Figure 4.5: The block classification indicative areas

four areas are denoted by the symbols DC, L, E, and H respectively. The groups of DCT coefficients that most strongly indicate the presence of a vertical, horizontal, or diagonal edge are also marked by three dotted lines respectively. For instance, as shown in figure 4.5, the sum of $L + E$ represents the overall edge strength in the block since the areas L and E include all of the indicative areas of an edge.

Three general block classes are defined for classification:

EDGE Blocks that contain a clear edge as the primary feature.

TEXTURE Blocks that contain a lot of complex spatial activities.

PLAIN Blocks that are generally smooth, with few spatial activities.

Figure 4.7 shows the block diagram of the proposed classification algorithm. Based on experimental findings it has been observed that high magnitudes in the ratios $\frac{L+E}{H}$ and $\frac{L}{E}$ strongly indicate the presence of an edge. As discussed previously, the sum $L+E$ represents the edge strength. The value H, which indicates the strength of the medium and high frequency coefficients, provides a measure of the 'textureness' of the block. Thus the ratio $\frac{L+E}{H}$ provides a relative measure of the strength of the two factors and a high value indicates that an edge is present.

The ratio $\frac{L}{E}$ is also found to provide a very good edge strength measurement for a block with an edge that is more diagonally oriented. Two example edge blocks from

(a) (b)

Figure 4.6: 8×8 blocks from the image *sail*

the test image *sail* are shown in figure 4.6. The ratios $\{\frac{L}{E}, \frac{L+E}{H}\}$ of block (a) of figure 4.6 are $\{1.7, 4.7\}$ and those of block (b) are $\{3.3, 1.9\}$ respectively. It can be observed that block (a) contains a horizontal edge and the ratio $\frac{L+E}{H} = 4.7$ correctly reflects that an edge is present. In block (b), the edge has a somewhat more tilted, or diagonal orientation. The value of H, which reflects the high frequency coefficients, is thus higher and the ratio $\frac{L+E}{H} = 1.9$ does not suggest the presence of a strong edge. However the ratio $\frac{L}{E} = 3.3$ is able to provide a strong indication of an edge since block (b) still contains a substantial amount of smooth areas on the two sides of the edge. Thus the the strength of the low spatial frequencies, or the value of L, is relatively high. In fact the ratio $\frac{L}{E}$ is also a perceptually significant measure as the human eye is very sensitive to the kind of sharp, clear edges on a mostly smooth background that $\frac{L}{E}$ indicates.

The classification of the TEXTURE class is relatively simpler and it is performed by investigation of the value of $E + H$. If the value is high, and the block does not satisfy the criteria for the EDGE class, it is assigned to the TEXTURE class. For the classification of the PLAIN class, if a block's $E + H$ value is low, or the block is not classified to the EDGE or the TEXTURE class, it is assigned to the PLAIN class.

The actual classification procedure is illustrated in figure 4.7. It can be divided into five condition blocks and the blocks are explained as follows:

- *Condition A:* If $E + H$ is smaller than or equal to μ_1, the block is classified as PLAIN. Otherwise it will be further tested. μ_1 is set to be 125 in this project.

- *Condition B:* It was experimentally decided that for a block with high spatial activities, an edge can still be present but the thresholds for detecting the edge

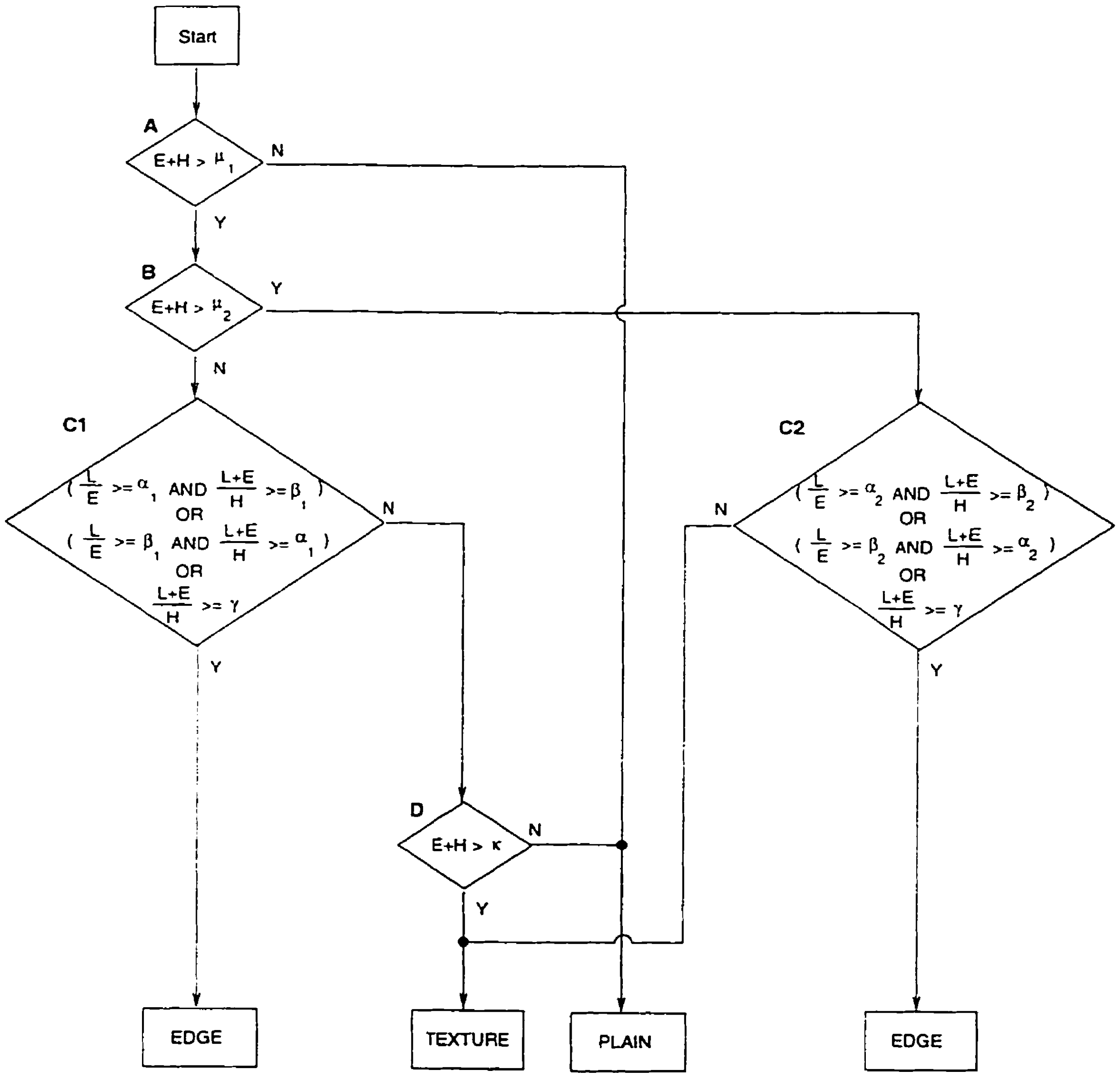

Start
A
E+H > μ1
N
Y
B
E+H > μ2
Y
N
C1
(L/E >= α1 AND (L+E)/H >= β1)
OR
(L/E >= β1 AND (L+E)/H >= α1)
OR
(L+E)/H >= γ
N
Y
C2
(L/E >= α2 AND (L+E)/H >= β2)
OR
(L/E >= β2 AND (L+E)/H >= α2)
OR
(L+E)/H >= γ
N
Y
D
E+H > κ
N
Y
EDGE
TEXTURE
PLAIN
EDGE

Figure 4.7: The block classification algorithm

will need to be lowered as compared to the thresholds for a block with moderate or low spatial activities. Thus two sets of EDGE thresholds are designed. If $E + H > \mu_2$ is false, condition C1 will be tested; otherwise condition C2 will be used. The value of μ_2 is set to be 900.

- *Conditions C1 and C2:* These two conditions are for the detection of the edge blocks. The ratios $\{\frac{L}{E}, \frac{L+E}{H}\}$ are the primary indicators of the presence of an edge. The higher thresholds of $\{\alpha_1, \beta_1\}$ of condition C1 are set to be $\{2.3, 1.6\}$. $\{\alpha_2, \beta_2\}$ of condition C2 are set to be $\{1.4, 1.1\}$.

  For example, in condition C1, the ratios $\{\frac{L}{E}, \frac{L+E}{H}\}$ need to be greater than either $\{\alpha_1, \beta_1\}$ or $\{\beta_1, \alpha_1\}$ for a block to be assigned to the EDGE class. A very high value in the ratio $\frac{L+E}{H}$ is also found to be sufficient as indication of an edge. So if $\frac{L+E}{H} > \gamma$ in condition C1 or C2, the EDGE class will also be assigned. γ is set to be 4.

- *Condition D:* If a block does not satisfy condition C2, it is assigned to the TEXTURE class. If a block does not satisfy condition C1, it is tested with the condition $E+H > \kappa$. If the outcome is true, the block contains sufficient texture activities and is assigned to the TEXTURE class, otherwise it is assigned to the PLAIN class. κ is set to be 290 in this project.

Locally adaptive classification correction

The edge detection thresholds of the classification algorithm are designed to be as reasonably sensitive as possible. However there might be misclassified edge blocks that should be texture blocks in reality. A simple locally adaptive correction scheme is used to re-classify the possibly misclassified blocks back to texture blocks using perceptual criteria. Figure 4.8 shows two scenarios when the correction is performed. From the figure, T represents texture block and E represents edge block. If one of the scenarios in figure 4.8 happens then the edge block in the scenario, which is the

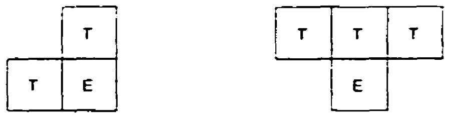


Figure 4.8: Locally adaptive classification correction

current block, is re-classified as a texture block. For causality,[2] only the blocks above and to the left of the current block are examined. The main rationale for performing the correction is that an edge block that is surrounded by texture blocks as shown in figure 4.8 is likely a misclassified block in a mostly textured region. Moreover, as discussed before, in general the human eye is more sensitive to a clear edge on a plain background. Thus the perceptual importance of the edge block in question, and the penalty against an incorrect re-classification is small as a result. The newly classified texture block will be assigned the lowest masking factor for the *TEXTURE* class (to be calculated and shown in table 4.1) to avoid over-harsh quantization.

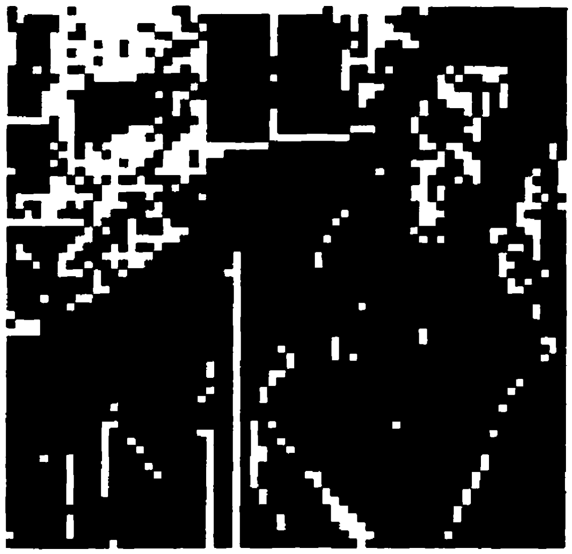
Classification examples

Figures 4.9 to 4.12 present some test images and their corresponding block classification maps, where the black colour represents the plain blocks, gray represents the texture blocks, white represents the edge blocks, and the darker the gray blocks, the higher the texture activities. The result for the image *lenna* is presented separately in chapter 5.

It can be observed that the algorithm is quite successful in differentiating the edge and texture blocks from the plain blocks. The algorithm's ability to detect edges is the main focus since the detection of texture is relatively simple - the sensitivity of the algorithm to texture can be modified easily by adjusting the parameter κ. The image *houses* is most challenging since it contains a lot of short, low contrast edges that are close to each other and are near the windows and roof-tops, which also contain some complex activities. However these edges are likely not very perceptually significant and texture masking with nearby features will probably mask the errors. In general

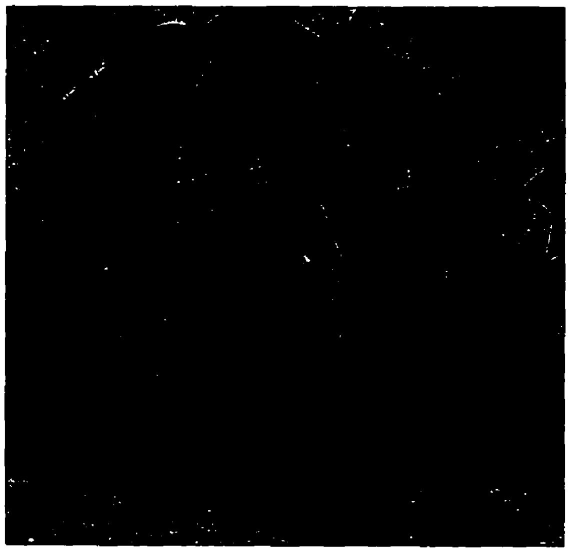
[2]In the current JPEG implementation, the processing of the 8×8 image blocks is from top to bottom, and left to right.

(a)

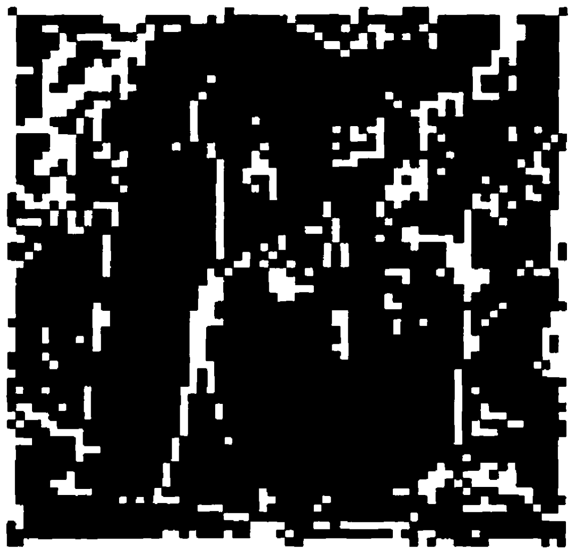


(b)

Figure 4.9: (a) *barbara*, (b) block classification



(a)



(b)

Figure 4.10: (a) *peppers*, (b) block classification

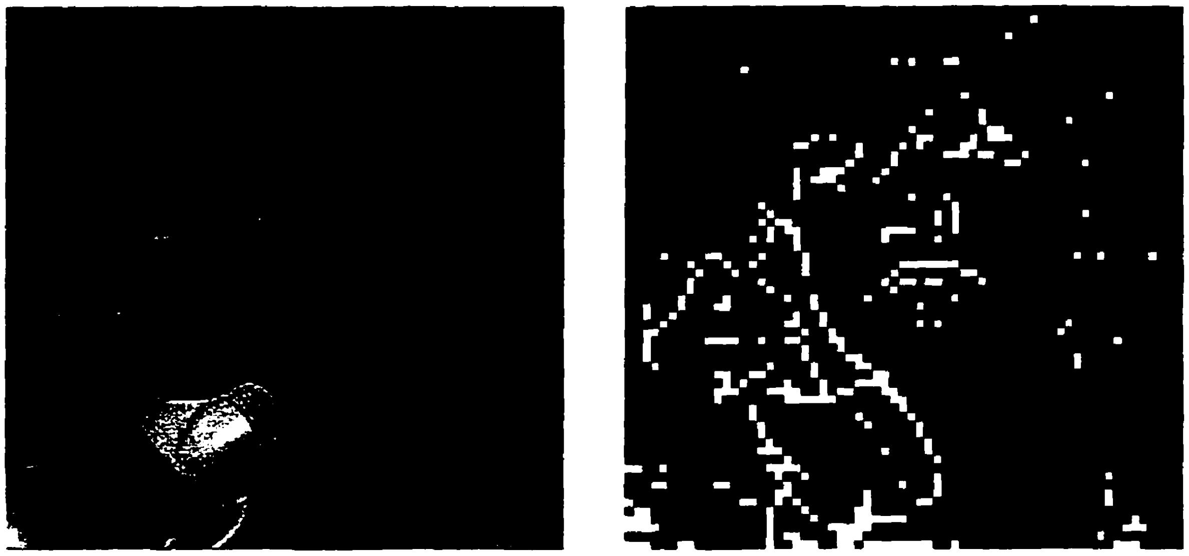

(a) (b)

Figure 4.11: (a) *susie*, (b) block classification

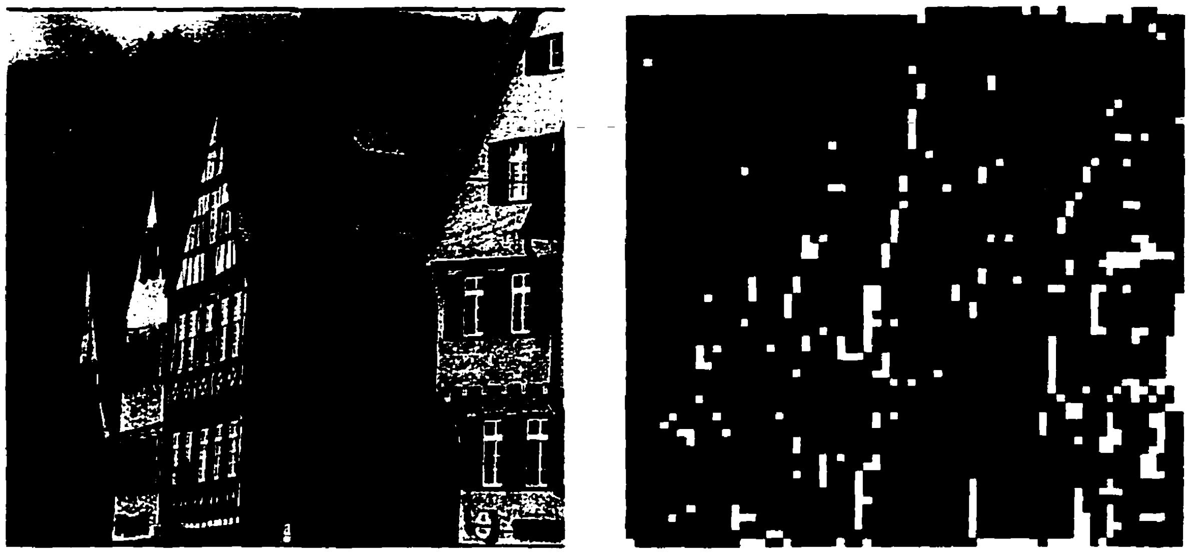

(a) (b)

Figure 4.12: (a) *houses*, (b) block classification

it can be observed that most of the clear edges in the images *barbara*, *peppers*, and *susie* are detected correctly and the texture areas are also well represented. Thus the results verify the accuracy of the classification algorithm.

4.2.2.2 Calculation of the texture masking factor

The distortion sensitivity varies among different block classes. In general, the sensitivity decreases in the order of plain, edge, and texture [42] and the texture masking factors are calculated independently for each class.

The texture blocks

The texture blocks can be quantized most coarsely. In fact they provide the majority of the redundancy removal in the texture masking model in general. The local adaptation depends on the local texture energy, which is approximated by the sum $E + H$, the L1 norm of the medium to high frequency AC coefficients. Using the linear threshold elevation model, the texture masking factor is derived as a function of the approximated texture energy:

$$TextureMask(k) = (MaxTextureElevation - 1) \times \frac{TexEnergy(k) - MinEnergy}{MaxEnergy - MinEnergy} + 1. \tag{4.7}$$

where $TexEnergy(k)$ is the local texture energy of the kth block, and $MaxEnergy$ and $MinEnergy$ represent the maximum and minimum energy of the texture blocks respectively. The parameter ***MaxTextureElevation*** can be used to scale the amount of texture elevation and is a convenient tool for fine-tuning the extent of the additional quantization introduced in the texture masking model. The parameters $MaxEnergy$ and $MinEnergy$ are determined to be 1800 and 290 respectively.

As an example, for $MaxTextureElevation = 2.25$, the masking factors for the $TEXTURE$ class are shown in table 4.1.

Texture Energy $(E+H)$	TextureMask
290-400	1.125
400-500	1.125
500-600	1.25
600-700	1.25
700-800	1.375
800-900	1.5
900-1100	1.625
1100-1300	1.75
1300-1600	2
>1600	2.25

Table 4.1: Example texture masking factors for the $TEXTURE$ class

The plain blocks

The plain blocks are the most sensitive to distortion and thus the $TextureMask(k)$ factor for the $PLAIN$ class is one, which means that no elevation of the global quantization matrix is performed.

The edge blocks

The edge blocks have been reported to be more tolerant to distortion perceptually compared to plain blocks [43, 42]. Nevertheless, over-harsh quantization of an edge block may result in 'dirty' blocks along an edge that are very objectionable to the human eye. Thus only moderate amount of texture masking elevation is designed and table 4.2 presents the texture masking factors empirically designed for the $EDGE$ class. Note that for the $EDGE$ class, the texture energy is approximated by the sum

Texture Energy $(L+E)$	TextureMask
≤400	1.125
>400	1.25

Table 4.2: Example texture masking factors for the $EDGE$ class

$L+E$ instead of the sum $E+H$ as in the case for the $TEXTURE$ class.

4.2.3 The luminance masking model

Chapter 2 introduces the concept of luminance masking, which shows that the human eye's sensitivity to distortion depends on the local background luminance. The luminance masking model computes the $LuminanceMask(k)$ factor that provides the local luminance-adjusted multiplying factor for use in equation 4.5.

4.2.3.1 The luminance sensitivity subjective experiment

An informal subjective experiment was performed to determine the change in the distortion visibility for various background luminance, or grayscale values. The main purposes for performing the test were to verify the validity of Weber's law (equation 2.3) and to collect a set of reference luminance sensitivity data for use in this project.

In the experiment [28, 20], a uniformly distributed random noise of known maximum magnitude is added to or subtracted from the pixels in an 100×100 square area of a background image with a constant grayscale level, and of size 360×360 pixels. Figure 4.13 shows an example testing image used in the experiment. The maximum

Figure 4.13: The luminance masking subjective experiment

magnitude of the noise square is adjusted until the observer cannot reliably determine if the testing image contains the noise square or not. And the test is repeated to determine the *noise visibility threshold* for different background grayscale levels. The position of the noise square on the testing image is randomized to avoid observer anticipation of the noise features in a fixed position. This differs from other similar experiments [28, 20] in which the noise square is always placed in the center of the image.

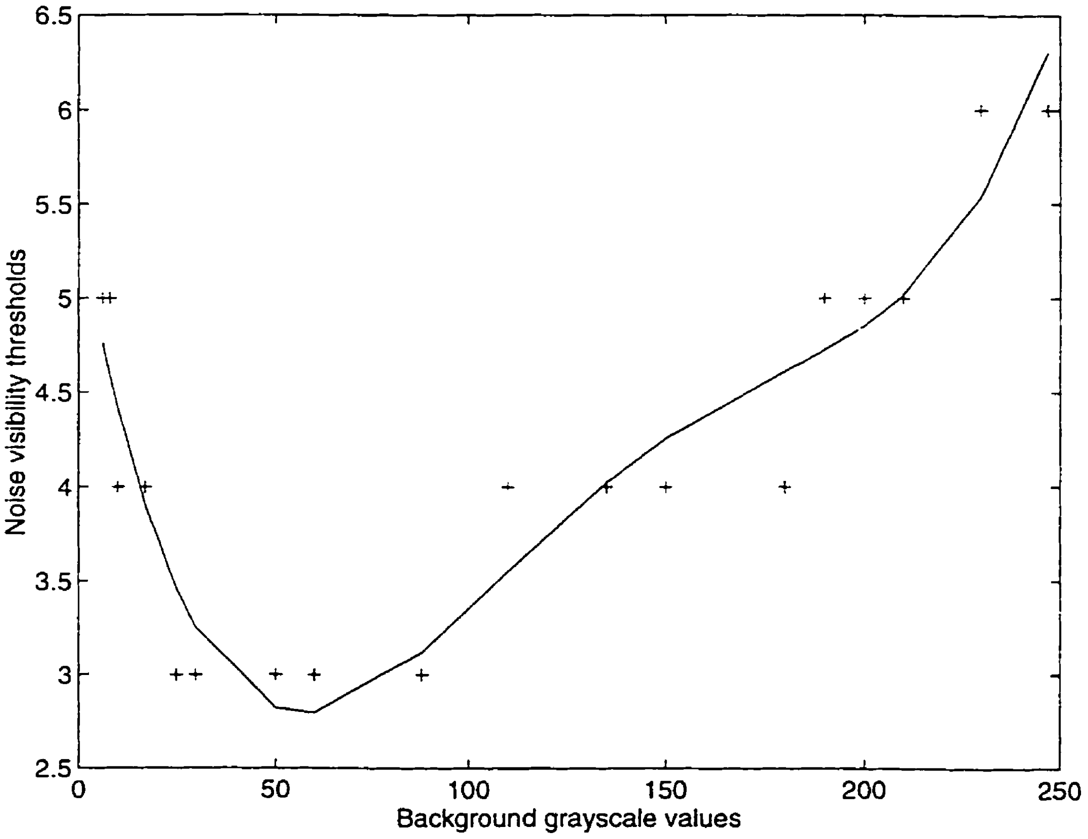


Figure 4.14: The noise visibility thresholds vs. background luminance

The experiment was conducted in a darkened room on a Pentium PC with a 24-bit colour Matrox Millenium graphics card and a 17" Viewsonic Optiquest V775 monitor. The test subject was the author, and the viewing distance was approximately six times the image height.

Figure 4.14 shows the graph of the measured noise visibility thresholds ΔL as a function of the background luminance L_B. The crosses are the measured thresholds and the curve is derived using the polynomial curve fitting function in the MATLAB program. The result in general is close to those reported in the literatures in a comparable form [1, 20]. And it confirms the well-known HVS property that the Weber fraction $\frac{\Delta L}{L_B}$ (equation 2.3) is approximately constant from medium-low to high luminance values and when L_B is low, $\frac{\Delta L}{L_B}$ starts to increase non-linearly with decreasing L_B [14, 21, 47, 48].

4.2.3.2 The influence of the viewing conditions

The viewing conditions should also be taken into consideration when interpreting the subjective testing results. The following factors are considered [49]:

Viewing distance It is obvious that the human eye's sensitivity to distortion decreases with increasing viewing distance. The viewing distance of six times the picture height follows the standard viewing conditions recommended for quality assessment of television images in Recommendation 500 by the *International Radio Consultative Committee (CCIR)* [50, 21, 51].

Ambient lighting It has been reported that the HVS is more sensitive in a dark room, and less sensitive when the lighting is brighter [49]. Thus the current subjective experiment, which was conducted in a slightly darker condition than it is in a normal office, offers a more conservative result with a relatively sensitive HVS response.

Monitor brightness The monitor brightness, or in more objective terms, the *luminance* or the *radiant energy*, can be measured by a digital Lux meter against the screen of the monitor [14, 49]. Most monitors also provide a switch for the user to adjust the local monitor brightness setting. The subjective experiment was informally conducted with different monitor brightness settings. It was discovered that for low brightness settings, the HVS was considerably less sensitive when the background luminance L_B was low (e.g. below 60, in the non-linear area in figure 4.14). For $L_B > 60$, which includes the range of most commonly used grayscale values, the effect of the change in the monitor brightness setting was limited. This finding also agrees with that reported in [49].

The monitor brightness setting for the current subjective test was about 75% (in a scale of 0%(lowest) to 100%(highest) for the testing monitor), which again provides a more conservative measurement as the HVS was more sensitive at this relatively high brightness setting.

Displaying system This factor refers to the influence of different video cards and monitors. The subjective experiment was also informally conducted with several different workstations and monitors; and the results in general are very similar with the current result. This also agrees with [49] which reports that the luminance masking measurements are virtually independent of the displaying system.

4.2.3.3 Calculation of the luminance masking factor

The idea of luminance masking is that the model can selectively scale up the quantization matrix where the added distortion will be masked by the local background luminance. The DC coefficient $F(0,0)$ represents the mean value of the input image block. Thus the DC coefficient of the luminance channel provides a readily available measure of the average background luminance of a block with which the $LuminanceMask(k)$ scaling factor in equation 4.5 can be calculated.

An important design decision is: at what DC luminance value(s) does the scale-up process begin? In the literatures, there are two common types of approaches to utilizing luminance masking in image coding: (1) Watson suggested the use of a scaling power function $(\frac{DC}{DC'})^a$, where DC' is the mean luminance of the display (or grayscale value 128 approximately) [48. 52]. (2) Other attempts utilized empirically derived luminance thresholds above or below which additional distortions are introduced locally [37, 53].

The main problem with using the above techniques for the current adaptive coding scheme is that they are not adaptive enough. Consider the scenario when the mean grayscale value of the whole picture is only about 90. Using the technique (1) above will result in under-utilization of the luminance masking model because most of the image blocks will have DC grayscale values below 128. Similarly, a bright picture with mean grayscale value of about 160 will result in over-aggressive quantization in the luminance masking model as the majority of the image blocks will have block DC values above 128.

In summary, the key to using luminance masking in a more robust way is to also take into account the mean grayscale value of each individual picture. An analysis of 24 colour and grayscale images in the digital signal processing laboratory has uncovered a wide range of average picture grayscale values[3] from 78 to 164, which further confirms the need for a more adaptive utilization of luminance masking.

The proposed luminance masking model will be divided into two parts and described in the next two sub-sections: a *linear modeling* part for medium to high background luminance, where Weber's law is used for the modeling; and a *non-linear approximation* part for low background luminance, where Weber's law can no longer be used. The main idea of the proposed scheme is to adjust the behaviour of the luminance masking model depending on the mean luminance of each individual image so that a reasonable amount of luminance masked distortion is ensured for every image.

The linear modeling for medium to high background luminance

Figure 4.15 shows the linear threshold elevation function that is an approximation of the Weber's law for blocks with medium to high average background luminance. The

[3]For colour images, the luminance component was used.

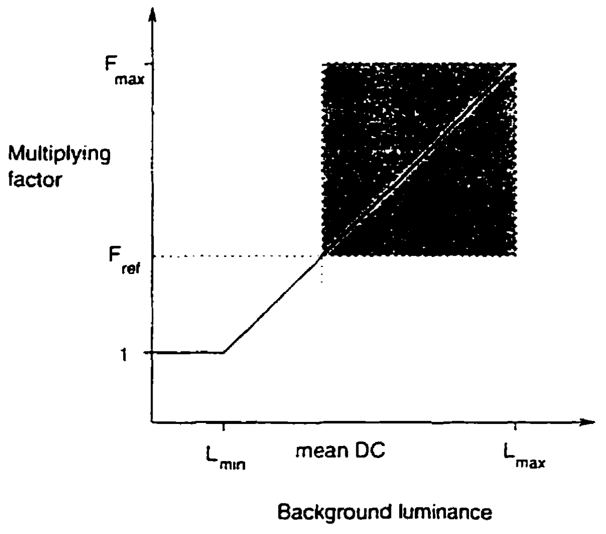


Figure 4.15: Luminance masking linear modeling

symbols L_{min} and L_{max} represent the range of the luminance values for the linear modeling and with respect to the subjective data from figure 4.14, they are chosen to be 90 and 255 respectively. F_{max} represents the *maximum luminance elevation factor*, or ***MaxLuminanceElevation*** . It controls the slope of the linear function and can be used to scale the extent of luminance masking like the parameter *MaxTextureElevation* for texture masking.

The *LuminanceMask(k)* factor is calculated as follows:

1. The mean grayscale value (*meanDC*) of the whole image is first calculated; the reference multiplying factor F_{ref} is then obtained from figure 4.15 using the *meanDC* value.

2. Only the blocks with DC values above *meanDC* will have a scaling factor ≥ 1. The active area for luminance masking is shown in the gray area in figure 4.15, and the luminance masking factor for the kth block, $DC(k) > meanDC$, is given by equation 4.8 below:

$$LuminanceMask(k) = (F_{max} - F_{ref}) \times \frac{DC(k) - meanDC}{L_{max} - meanDC} + 1, \tag{4.8}$$

and if $DC(k)$ is smaller than $meanDC$,

$$LuminanceMask(k) = 1,\ L_{min} < DC(k) \leq meanDC. \qquad (4.9)$$

The $LuminanceMask(k)$ factors are image-dependent as the calculation is based on the mean luminance value of each image. An example table of the luminance masking factors for medium to high background luminance will be presented in chapter 5 for the image *lenna*.

The non-linear approximation for low background luminance

The low luminance areas of the image (grayscale $\leq L_{min} = 90$) need to be treated carefully because of the following factors: (1) The relationship between the visibility threshold and the background luminance: $\frac{\Delta L}{L_B}$ becomes non-linear and is hard to model. (2) Except for the very dark areas, the low luminance areas are in general most sensitive to errors.

A conservative luminance masking scheme is thus designed in light of the sensitivity and non-linearity problems for low background luminance. Table 4.3 presents a set of empirically designed multiplying factors that are based on the experimental result from figure 4.14. As described in sub-section 4.2.3.2, the distortion visibility

Block DC values	LuminanceMask
0-15	1.25
15-25	1.125
25-90	1

Table 4.3: The luminance masking factors for low background luminance

in low luminance areas is very sensitive to the local monitor brightness setting. Thus the parameters presented here, which are designed under a relatively high monitor brightness setting, only provide a conservative degree of luminance masking and under informal testings have been shown to perform well for a wide range of images and different viewing conditions.

4.2.4 Processing for the subsampled chrominance channels

For colour images, the $m(k)$ multiplier map is generated from the data in the luminance channel. However, since the two chrominance channels are subsampled two to one both horizontally and vertically, the multiplier map cannot be used directly for the chrominance data. This section presents a simple algorithm that is used to adapt the multiplier map for use in the chrominance channels.

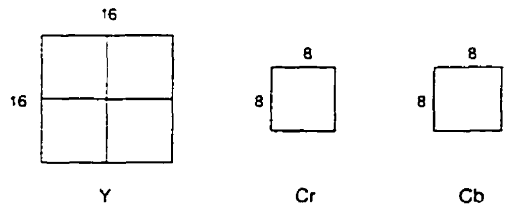


Figure 4.16: A 16 × 16 area: four Y blocks, one Cr block, one Cb block

Figure 4.16 shows a 16 pixel by 16 pixel area of a colour image separated into three colour channels, using the YCrCb colour space. Because of subsampling, the chrominance channels contain in reality only one 8 × 8 block for each chrominance channel in the 16 × 16 area. In other words, there are four luminance-generated $m(k)$ values in each 16 × 16 area and the problem is to find a coresponding *chrominance multiplier* m_c using the four luminance $m(k)$'s.

The generation of the chrominance multiplier cannot be dependent on the information from the perceptual model, e.g. the block classification results, because the same process also has to be done in the decoder and the perceptual model is implemented in the encoder level only. Thus the only available data are the luminance multiplier map, and the chrominance multiplier generation algorithm is illustrated in figure 4.17.

The algorithm is a generalization of a similar procedure presented in [46]. The main idea is to inspect the four luminance $m(k)$ values and count the number of m(k)'s with the value one (condition A), or the number of blocks with no local QM elevation. If the count is bigger than one, the percetual model indicates that at least two of the luminance blocks are sensitive to error and should not be further quantized. Thus the chrominance multiplier m_c is also assigned to be one, giving priority to the sensitive area. If the count of $m(k)$'s equal to one is one or less, at least three of the luminance blocks have $m(k)$'s bigger than one. A majority rule is used and m_c is assigned to the smallest $m(k)$. $m(k) \neq 1, k = 1,2,3,4$. Note that the algorithm will always assign m_c the value of the lowest or the second lowest $m(k)$ out of four possible luminance $m(k)$'s. This reduces the chance of the chrominance blocks being over-quantized when they are located near a strong luminance edge.

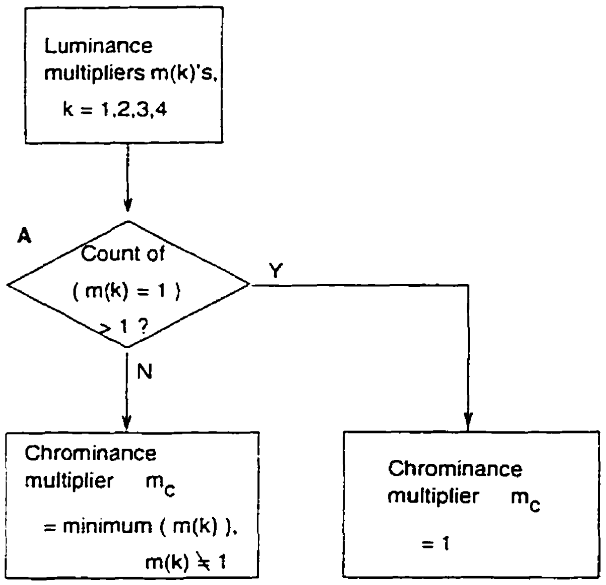

Luminance multipliers m(k)'s, k = 1,2,3,4
A
Count of (m(k) = 1) ≥ 1 ?
Y
N
Chrominance multiplier m_C = minimum (m(k)), m(k) ≠ 1
Chrominance multiplier m_C = 1

Figure 4.17: Generation of the chrominance multiplier

4.3 Coding of the overhead information

The overhead mode (figure 4.2) requires that the adaptive multiplier map be sent as overhead information with the compressed image data for decoding. Encoding of the overhead information is thus necessary to help reduce the overhead cost. This section discusses two encoding methods that are examined in this project.

The standard coding method

The predominantly slowly varying nature of JPEG's target continuous-tone images suggests that neighbouring blocks will likely have identical local multiplying factors

$m(k)$'s [54, 52]. The method discussed in this section is a simple technique employed in most video coding standards for encoding of the local parameters [14].

The technique is a variant of differential coding. It utilizes a single-bit status bit that monitors if the current block requires a new multiplying factor than the one used by the previous block, and a 5-bit index that specifies the value of the multiplying factor. If a new multiplying factor is needed, the block overhead costs are 6 bits: the status bit pluses the 5-bit new index; otherwise the coder just set the status bit and the cost is 1 bit.[4] Table 4.4 summarizes the overhead coding cost for using the standard overhead coding method.

Block status	Code description	Code size
No change from previous	status bit	1
New multiplier	status bit, new index	6

Table 4.4: Overhead coding - Standard

A Lempel-Ziv estimation of the overhead cost

The standard overhead coding scheme makes a simple assumption that consecutive image blocks are more likely to use the same multiplier. The coding parameters are independent of the statistics of the input multiplier map and experimentation with different images has shown that the compression ratio achieved by using the standard coding method over plain transmission of the uncompressed multiplier map ranges from about 1.7 to 3 times.

For medium and high bit-rate image coding systems, the overhead cost is generally insignificant compared to the total compressed image data. However, when high compression ratio is desired, the overhead cost might actually account for a significant contribution to the total cost and it is beneficial to also examine more optimal ways of encoding the overhead information [54].

[4]In practice, the status bit is generally only one part of a status byte (or bytes) that also contains other status information. The status byte might also be variable length coded, so the 1-bit size of the status bit is only an approximation which might vary slightly across different pictures.[14]

The goal is to find an adaptive overhead coding scheme that can adapt to the input statistics and thus generate a more compact representation of the multiplier map. A universal coding algorithm. Lempel-Ziv coding (chapter 2), is chosen because of its speed, the availability of programming source, and its close approximation of the source entropy rate [5, 55].

The *Lempel-Ziv-Welch (LZW)* algorithm, which is a variation of Lempel-Ziv coding, is used [56]. The programming source code is obtained from the GIF file format encoder/decoder functions[5] that are parts of the JPEG software library used in this project [13]. In general, for coding the overhead information, the gain in compression ratio ranges from 16 to 26% compared to using the input-independent standard codes.

[5]The GIF file format uses the LZW method to perform lossless image compression.

Chapter 5

Experimental results

5.1 Introduction and set-up of the coder parameters

The methodology proposed by Safranek in [31, 29] is utilized in this project to examine the performance of the adaptive coder. The idea of the methodology is that both *qualitative* and *quantitative* measurements must be examined for a perceptual coding system. Qualitative measurements ensure that the introduction of the additional distortion by the adaptive coder does not lower the perceived quality of the coded image, when compared to the result using a non-adaptive baseline JPEG coder with the same quality factor. Additionally, quantitative measurements, for instance, bit-rate savings, are also needed to demonstrate the advantage of the adaptive method objectively.

In this project, the quantitative measurements will be described as the objective results; whereas the qualitative measurements will be described as the subjective results. Bit-rate savings and a comparative subjective test performed with a group of human subjects are the primary objective and subjective measures utilized respectively.

Table 5.1 summarizes the coder parameters examined in this project. The target application area for the adaptive coder is for high quality image compression. The

Descriptions	**Parameters**
Quality factor	72
Adaptive modes	JPEG-compatibility, Overhead
Overhead info coding	Standard, LZW
MaxTextureElevation	2.25
MaxLuminanceElevation	2

Table 5.1: The adaptive coder parameters

JPEG quality factor 72, which produces a bit-rate of 0.916 bit/pixel for the *lenna* grayscale image with the baseline JPEG coder, is used. This bit-rate has been reported to produce perceptually transparent quality for *lenna* in a previous work [57]. Using the same perceptual model parameters, the performance of adaptive coding both with overhead (overhead mode) and without overhead (JPEG compatibility mode) will be examined. For the coding of the overhead information, the standard coding method and Lempel-Ziv coding will be investigated. Although the adaptive coder is designed for the perceptually lossless quality level, this quality level is a rather subjective measure and it is instructive to also examine the coder's performance at other bit-rates. The results for varying bit-rates will be presented in section 5.4.

The parameter *MaxTextureElevation* is used to scale the extent of texture masking in the perceptual model. The higher the parameter, the more the compression in the texture area but the lower the quality. The value 2.25 has been found to provide a good balance between compression performance and perceptual quality for the test images compressed using the two adaptive modes. The parameter *MaxLuminanceElevation* for luminance masking is chosen to be 2. This choice is based on the subjective test results from figure 4.14, where the approximated maximum noise visibility threshold, 6, is two times the minimum threshold 3.

Table 5.2 sums up the JPEG system parameters used in this project. The 4:2:0

Descriptions	**Parameters**
Colour space	YCrCb
Colour subsampling	4:2:0
Quantization matrices	JPEG standard example matrices
Huffman tables	Customized

Table 5.2: The JPEG system parameters

colour subsampling scheme requires that the colour spaces Cr and Cb be subsampled two pixels to one both horizontally and vertically. The JPEG example quantization matrices are shown in figures 3.2 and 3.3. Customized Huffman tables, which are image-dependent, have been reported to provide improvements over the example Huffman tables listed in the standard by a few percent in bit-rate [13]. The JPEG system parameters are listed here for completeness. They remain the same throughout the chapter as the emphasis is on the comparison between the baseline JPEG and the adaptive JPEG coders.

Eighteen 512 $\times$ 512 grayscale and colour images are used in this project. They consist of both standard test images and high quality Kodak images[1], chosen to provide a variety of different examples of JPEG's target continuous-tone photographic images. The images are shown in thumbnail forms in figures 5.1 to 5.4.

[1]The images can be found on the internet at ftp://ipl.rpi.edu/pub/image/still.

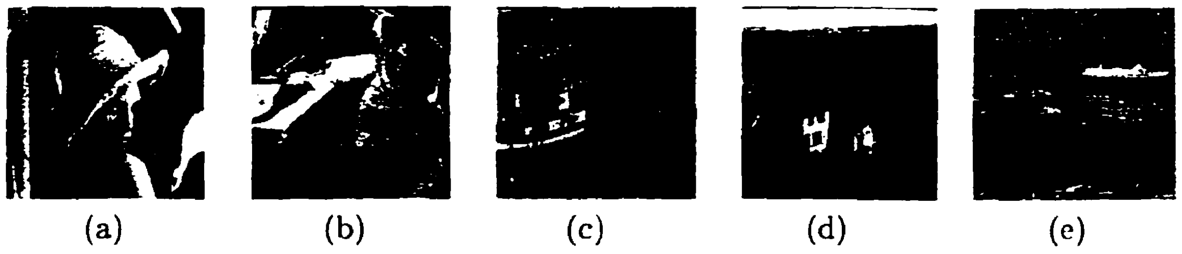


Figure 5.1: (a) *lenna*, (b) *barbara*, (c) *boats*, (d) *goldhill*, (e) *harbor*

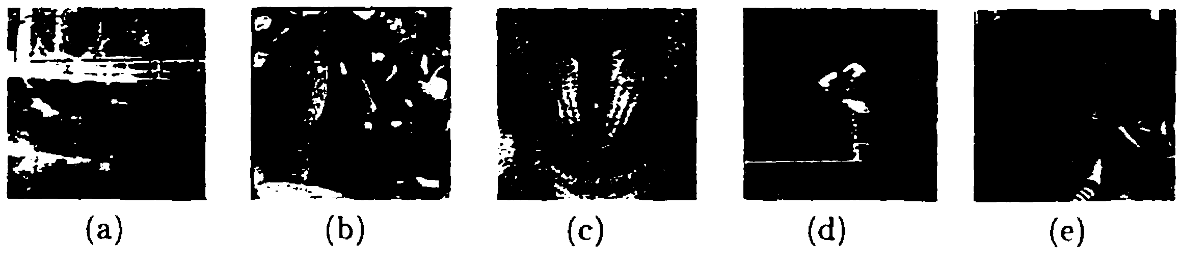


Figure 5.2: (a) *bridge*, (b) *peppers*, (c) *mandrill*, (d) *flower*, (e) *lady*

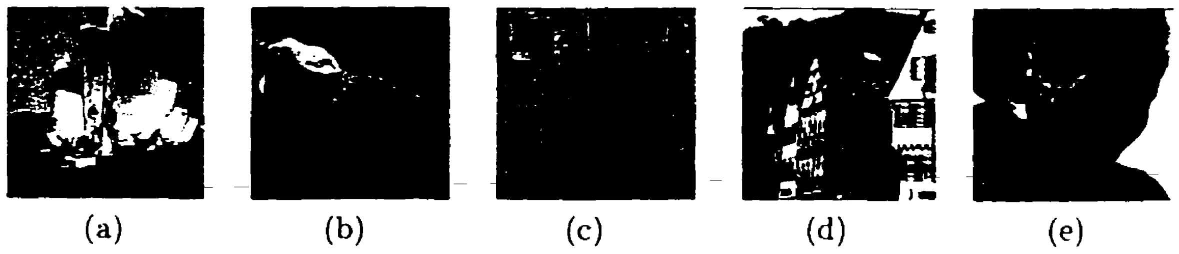


Figure 5.3: (a) *sail*, (b) *hats*, (c) *windows*, (d) *houses*, (e) *girl*

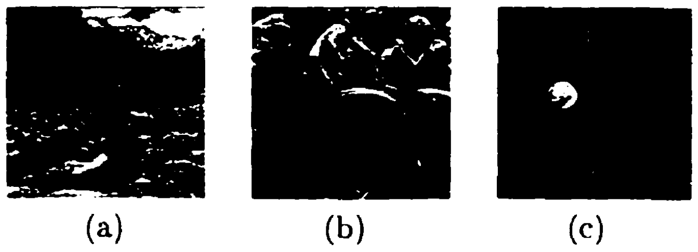


Figure 5.4: (a) *river*, (b) *motorbike*, (c) *lock*

5.2 A coding example with the image *lenna*

The popular grayscale *lenna* image is used to illustrate the operation of the adaptive coder. Figure 5.5 shows the original *lenna* image. Figure 5.6 shows the results of performing block classification on *lenna*, where the black colour represents the plain blocks; gray represents the texture blocks; white represents the edge blocks; and the darker the texture blocks, the higher the texture activities. It can be observed that the algorithm successfully locates the feather areas of *lenna*'s hat as texture areas. The edges are also well represented by the edge blocks.

The performance of the perceptual model is also examined. Figure 5.7 shows the texture masking multiplier map for *lenna*, where the black colour represents the lowest multiplier value of one, i.e. no change from the global quantization matrix. The lighter the blocks, the higher the multiplier values. It can be observed that the lightest blocks are mostly located around the feather areas of *lenna's* hat, where the textured, complex activities are concentrated.

As explained in chapter 4, the values of the luminance masking multiplying factors for medium to high background luminance depend on the average grayscale value (mean DC) of each individual image. The mean DC value of *lenna* is 123. Using the procedure described in section 4.2.3.3, the luminance masking multiplying factors for background luminance > 90 are calculated and shown in table 5.3. Table 4.3 in section 4.2.3.3 lists the multipliers for background luminance below 90.

Block DC values	LuminanceMask
90-140	1
140-160	1.125
160-180	1.25
180-200	1.375
200-220	1.50
220-240	1.625
240-255	1.75

Table 5.3: The LuminanceMask factors for medium to high background luminance for *lenna*

Figure 5.8 shows the luminance masking multiplier map for *lenna*. Again the

Figure 5.5: The original *lenna* image

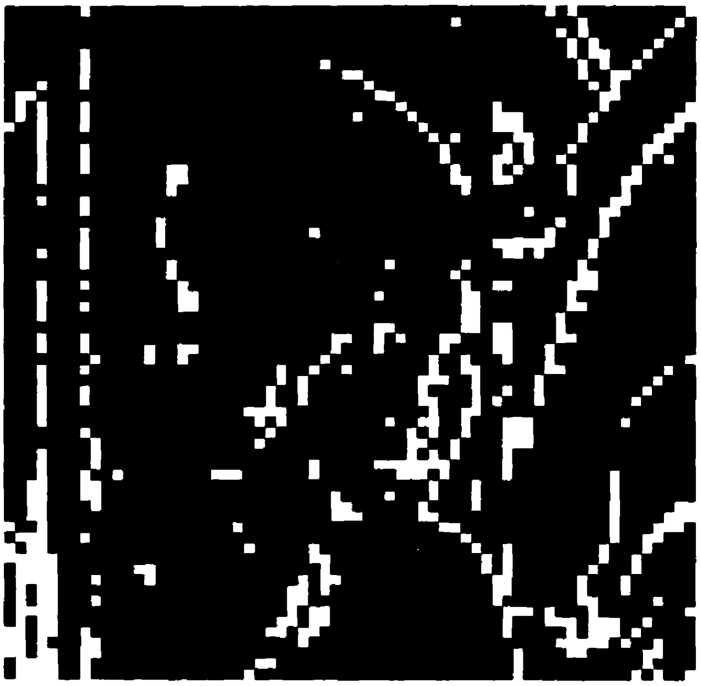


Figure 5.6: Block classification of the *lenna* image

Figure 5.7: The texture masking multiplier map for *lenna*

Figure 5.8: The luminance masking multiplier map for *lenna*

lighter the blocks, the higher the multiplier values. It can be observed that the model is able to locate the bright areas of the *lenna* image quite successfully. The highest LuminanceMask value in table 5.3 is 1.75, which is smaller than 2, the value of the *MaxLuminanceElevation* factor shown in table 5.1. This is because the *MaxLuminanceElevation* factor is designed for the whole dynamic range of medium to high background luminance from 90 to 255. The mean DC value of 123 allows luminance masking only for background luminance from 123 to 255, a smaller dynamic range and hence lower elevation factors overall.

Figure 5.9 shows the final, combined multiplier map for *lenna*. Each entry in the combined multiplier map is just the product of the corresponding texture masking multiplier and luminance masking multiplier presented in the previous figures. The reconstructed *lenna* image, coded under the JPEG compatibility mode at the JPEG quality factor 72, is shown in figure 5.10.

5.3 Compression results for a fixed quality factor

The performance of the adaptive coder using a fixed JPEG quality factor (Q) 72 is examined in this section. The objective bit-rate savings performance will first be presented and the subjective results will follow.

5.3.1 Objective results

This section presents the bit-rate savings over baseline JPEG that can be achieved by the adaptive coder, under both JPEG compatibility and overhead modes. Tables 5.4 and 5.5 show the results for grayscale and colour images respectively. The bit-rates using baseline JPEG are also presented for reference purposes. The baseline JPEG bit-rate is a very good measure of the overall image complexity. For the same quality factor, the higher the compressed bit-rate, the more difficult the image is to compress, which in turn implies that the image is relatively complex and lacks statistical redundancy which can be exploited by baseline JPEG.

As an example, an uncompressed grayscale image requires 8 bits per pixel. Thus

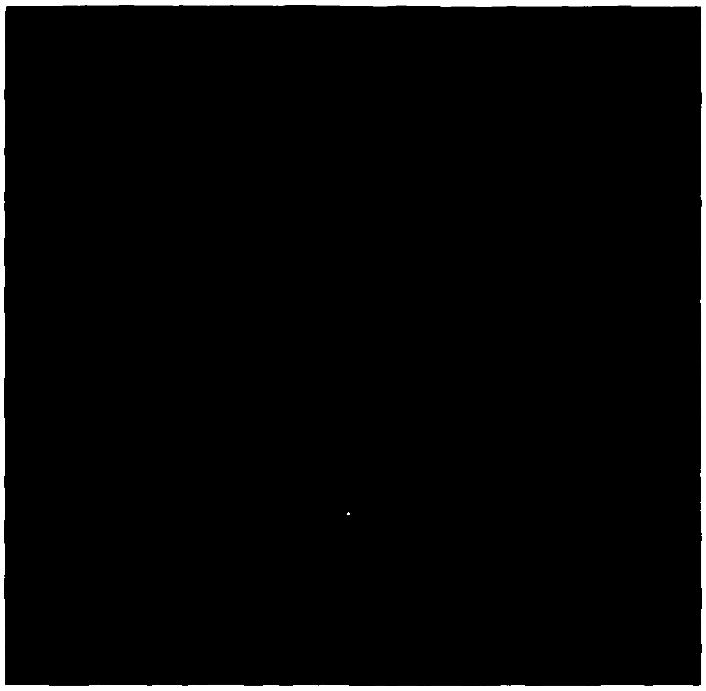


Figure 5.9: The combined multiplier map for *lenna*

Figure 5.10: The reconstructed *lenna* image, JPEG compatibility mode

Image	Baseline JPEG bit-rate (bit/pixel)	JPEG mode bit-rate savings	Overhead mode	
			savings (no overhead)	savings (overhead(LZW))
lenna(g)	0.92	5.1%	9.4%	5.5%
barbara	1.3	6.8%	15%	11.2%
boats	0.97	5%	9.7%	6.4%
goldhill	1.3	6.2%	11%	8.2%
harbor	1.3	9%	18%	14%
bridge	1.8	12%	21%	18%
Average		7.4%	14%	11%

Table 5.4: The bit-rate savings over Baseline JPEG (grayscale)

Image	Baseline JPEG bit-rate (bit/pixel)	JPEG mode bit-rate savings	Overhead mode	
			savings (no overhead)	savings (overhead(LZW))
lenna(c)	1.05	5.2%	9.5%	6%
peppers	1.11	6%	11%	7.5%
mandrill	2.19	13%	23%	21%
flower	1.08	5.2%	10%	6.5%
lady	1.02	5%	9%	5.8%
sail	1.08	7.1%	12%	9%
hats	0.84	7.1%	14%	10%
windows	1.77	11%	19%	16%
houses	1.86	10%	20%	17%
girl	1.05	6%	11%	8%
river	2.16	13%	25%	22%
motorbike	2.01	8%	17%	14%
lock	1.02	3.4%	7.2%	4.5%
Average		7.7%	15%	11.3%

Table 5.5: The bit-rate savings over Baseline JPEG (colour)

for the grayscale *lenna* image, the baseline JPEG bit-rate of 0.92 bit/pixel implies that the compression ratio is $\frac{8}{0.92}$, or approximately 8.7 times and the lower the bit-rate, the higher the compression ratio. Note that the compression ratio for the colour images are in reality much higher than those for the grayscale images because an uncompressed colour image needs 24 bit/pixel representation as opposed to 8 bit/pixel for a grayscale image. The higher compression ratio for colour images is mainly due to the sub-sampling of the two chrominance channels and the use of the separate chrominance quantization matrix, which has higher quantization step sizes than its luminance counterpart.

It can be observed that using the same perceptual model parameters, the bit-rate savings achieved by the JPEG compatibility mode range from about 3% to 13%, while for the overhead mode (with overhead information coded with the LZW method), the range is from 5% to 22%.

In general, the results show that the harder the image to compress (high baseline JPEG compressed bit-rate), the better the adaptive coder performs. Figure 5.11

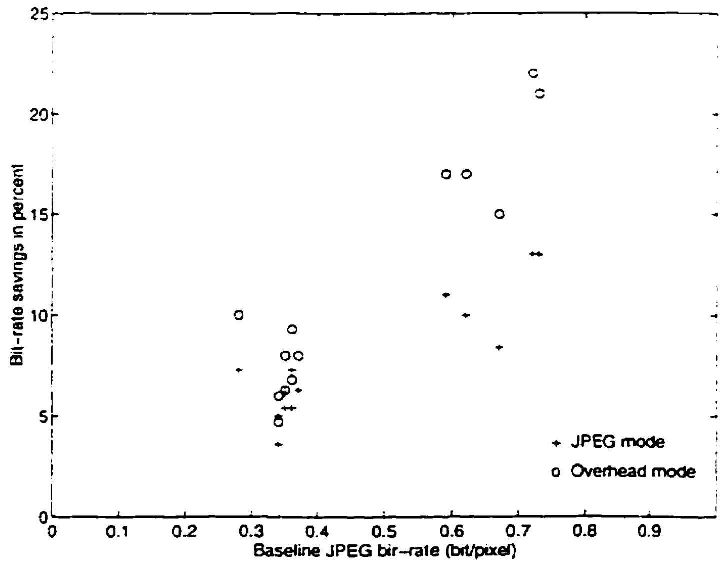


Figure 5.11: Bit-rate savings vs. baseline JPEG bit-rate (colour images)

illustrates that indeed the adaptive coder has better performance for images that do not compress well with baseline JPEG (complex images with baseline JPEG bit-rates over 1.5 bit/pixel). This confirms a similar finding reported in [31]. The result is actually not surprising since those hard-to-compress images often contain a lot of high activity areas that contain large DCT coefficients that are difficult to quantize. The texture masking property of the HVS suggests that distortions in these high activity areas are also hard to notice. Thus an important key to the adaptive coder's performance is the texture masking model which is able to locate the high activity areas and scale up the quantization matrix accordingly for improved compression. It can also be observed that for smooth images with baseline JPEG bit-rates of about 1 bit/pixel or below, the coder's performance is more limited and is in the range of about 5 to 10%. In general, the more complex the image, the harder it is to compress using baseline JPEG, and the better the adaptive coder performs.

In fact a similar observation can be extended for the luminance masking model. which more coarsely quantizes blocks that have much brighter or darker average luminance than the mean luminance of the image. Thus an image with a broad and relatively flat luminance histogram will benefit greatly from luminance masking in the perceptual model.

Impact of the overhead information

The savings with the overhead mode *without* the overhead information are also presented in tables 5.4 and 5.5. Note that an image compressed with the overhead mode cannot be decoded correctly without the overhead information. However, the savings in this case provide a more accurate indication of how much more compression can be achieved by using full adaptive quantization instead of just adaptive thresholding as in the case in the JPEG compatibility mode. It is clear from the tables that performing full adaptive quantization can almost *double* the raw bit-rate savings when the overhead information is not included. This section analyzes briefly the impact of the overhead information on the final bit-rate savings in the overhead mode.

The overhead information in the overhead mode generally constitutes about 3 to 4% of the total bit-rate (savings without overhead minus savings with overhead). Although for all of the test images, the overhead mode out-performs the JPEG compatibility mode at Q = 72, the benefit of using the overhead mode diminishes considerably (see figure 5.11) when the images are relatively smooth since there is not enough perceptual redundancy that can be exploited as explained in the previous section. For these low complexity images, the overhead information required offset most of the limited additional savings that could be realized by using the overhead mode.

The test images in this project are all 512 × 512 images. The corresponding size of the uncompressed multiplier map is thus $\frac{512}{8} \times \frac{512}{8} = 4096$ bytes. Table 5.6 shows

Image	Standard overhead size (bytes)	LZW overhead	
		size (bytes)	improvements
lenna(c)	1631	1199	26%
barbara	1901	1578	17%
boats	1449	1035	29%
harbor	1810	1476	18%
peppers	1616	1299	20%
mandrill	2117	1682	21%
flower	1613	1235	23%
lady	1427	1104	23%
motorbike	2414	2039	16%

Table 5.6: Overhead coding - Standard vs LZW

the size of the coded overhead with standard coding and LZW coding for a sub-group of test images. It can be observed that LZW coding consistently provides at least 16% improvements over the input-independent standard coding method. Thus for theoretical comparison purposes, this project chooses the LZW method for overhead coding in the overhead mode to more accurately model and analyze the performance differences between adaptive coding with overhead (overhead mode) and without overhead (JPEG compatibility mode).

5.3.2 Subjective results

5.3.2.1 Description of the subjective test

A subjective test was performed to compare the subjective quality between the baseline JPEG coded images and the A-JPEG coded images. The seven-grade comparison scale (table 5.7) from the CCIR Recommendation 500 [50, 51] was used. The test was conducted on a Pentium-Pro PC with a 24-bit colour Matrox Millenium graphics card and a 17" Nanao Eizo FlexScan TX.C7S Trinitron monitor. The viewing distance was four times the picture height, which was closer than the recommended distance of six times [50, 21], and the testing room was under normal office lighting condition.

A group of 19 subjects were invited to participate in the test. Seven of the participants were members of the digital signal processing group. The rest of the participants were non-experts and they were mostly undergraduate and graduate students in the university. A test subset of eleven pictures was used.

Comparison scale	
+3	Much better
+2	Better
+1	Slightly better
0	The same
-1	Slightly worse
-2	Worse
-3	Much worse

Table 5.7: The CCIR comparison scale

The testing procedure was conducted as follows:

1. In each comparison the subject was simultaneously shown two images side by side on the monitor screen. One of them was the baseline JPEG image and the other was the adaptive JPEG image. There are two comparisons for each baseline JPEG image so that each of the two images produced by the two adaptive modes respectively will be compared once with the baseline JPEG image. The screen location (left or right), and the order of appearance of the adaptive coding modes (JPEG compatibility mode first, overhead mode second, or vice versa) were both randomized. The JPEG quality factor of 72 was used for all baseline JPEG and A-JPEG images in the test.

2. The subject was told that one of the images contained more distortion than the other and was asked to use the comparison scale in table 5.7 to evaluate the images. The subject was allowed to view the pictures for as long as he or

Image	JPEG mode result	Overhead mode result
lenna(c)	0.316	0.158
barbara	0.368	0.053
boats	0.000	0.158
peppers	0.316	0.421
goldhill	0.263	0.211
mandrill	0.263	0.053
flower	0.421	0.474
hats	0.211	0.526
lady	0.158	0.316
sail	0.158	0.000
motorbike	0.316	0.368
Average	0.254	0.249

Table 5.8: The subjective test results - by images

Subject No.	JPEG mode result	Overhead mode result
1	0.182	0.091
2	0.273	0.455
3	-0.182	0.182
4	0.636	0.455
5	0.000	0.455
6	0.091	0.000
7	0.091	0.182
8	0.818	0.273
9	0.727	0.727
10	0.000	0.091
11	0.545	0.455
12	0.182	0.273
13	0.636	0.000
14	0.727	0.364
15	0.273	0.182
16	-0.273	-0.182
17	-0.091	0.364
18	0.273	0.364
19	0.000	0.000
Average	0.254	0.249

Table 5.9: The subjective test results - by subjects

she desired. Moreover, the subject was encouraged to use the mouse to change the locations of the pictures on the screen to make sure that any perceived distortion was not due to possible non-uniformity across the monitor screen. However, manual zoom-in of the images was prohibited.

3. After the two comparisons for each baseline JPEG image were completed, the subjects were told if their choices of the 'better' images were indeed the less distorted baseline JPEG image. However they were not allowed to see the images again until after the whole test was finished.

The results of the subjective test are presented in tables 5.8 and 5.9. Table 5.8 shows the test images in their order of presentation in the test, i.e. *lenna*(c) (c represents colour) was always the first test image; and *motorbike* the last. A positive result means that the baseline JPEG image on average was rated 'better' to a certain degree (using the scale in table 5.7) than the A-JPEG image. The higher the value of the result, the more the baseline JPEG image was perceived 'better' than the A-JPEG image. Table 5.9 shows the results by subjects, where a positive result means that the subject on average rated the baseline JPEG images 'better' to a certain degree than the A-JPEG images.

5.3.2.2 Discussions

The low average results of 0.254 for the JPEG compatibility mode and 0.249 for the overhead mode show that for the test image set, the two adaptive modes both produce output that are essentially indistinguishable from those produced with baseline JPEG. Nevertheless the fact that the results are not closer to zero or even negative means that there is still some slight difference between a baseline JPEG image and an A-JPEG image, and on average viewers are a little more likely to prefer the less-distorted baseline JPEG image than the A-JPEG image.

The most puzzling implication from the test is that the subjective results for the two adaptive coding modes are almost identical - in fact the overhead mode was even rated a bit closer to baseline JPEG than the JPEG compatibility mode.

This is intriguing since by definition, using the same perceptual model parameters, the multiplier map overhead allows more aggressive quantization in the overhead mode than the adaptive thresholding operation of the JPEG compatibility mode (see section 4.1.1.3). The author, who is familiar with the types of distortion introduced by A-JPEG, has done the test several times (the result is not included because it is obviously biased). The overhead mode always achieves a higher subjective result than the JPEG compatibility mode and this was the expected outcome before the test was conducted.

An answer to the confusion between the two adaptive modes is that the human eye simply is not sensitive enough to be used as a reliable quality metric. It may be able to tell a baseline JPEG image from an A-JPEG one. But it will not be a successful metric for differentiating a *more distorted* A-JPEG image from another *less distorted* A-JPEG image. This raises the need for a reliable perceptual metric that can replace the traditional mean-square based metrics and also be computed objectively [58, 59].

Another possible answer to the above problem lies in some observers' inability to differentiate distortion from true image features. At least four or five subjects (subjects 13 and 16 in particular) had indicated that the distorted images looked 'sharper' to them and consequently they picked those images over the baseline JPEG images.[2] As the overhead mode images were the most distorted, they were most likely to be mistaken as 'better' than baseline JPEG images. In the subjective test, this created negative results and could pull down the overhead mode's overall result. (Interestingly, several subjects who obtained high results indicated that they picked the baseline JPEG images because the A-JPEG images appeared less sharp in some high details areas.)

The following summarizes other notable issues regarding the subjective test:

- The subjects' abilities to notice distortion vary greatly. Some subjects had sharp eyes and were able to see some differences between almost every pair of

[2]Subject 13 used to work in Sony Japan as a video engineer and he mentioned that the granularity introduced in the A-JPEG images was similar to the effect of increased picture sharpness in a television set.

images. Indeed the results obtained by subject 9 were even higher than those of the author, who were much more familiar with the images. Yet some other subjects were less successful and were not able to see any difference at all.

- In general most subjects agreed that the baseline JPEG images and the A-JPEG images were very close in quality and the comparison scale in table 5.7 was never extended beyond the subset {-1, 0, +1}. This is the main reason for the design of the relatively strict test where the user was allowed an unlimited viewing time, freedom to move the images around, and a close viewing distance of four times the picture height. Most viewers indeed require about thirty seconds to one minute before a decision can be made. Thus it is reasonable to expect that in a more casual viewing situation, the percentage of users noticing the difference between the baseline JPEG images and A-JPEG images will be even lower.

- The subjects were told if their picks were the baseline JPEG images after the two comparisons with each baseline JPEG image were finished. A few subjects were quick-learners and they were able to pick the baseline JPEG images much more accurately in the last few comparisons than in the beginning. They indicated this and voiced concerns that the results for the last few images might not be accurate. Nevertheless, from table 5.8, the results for the last few images do not show large deviations from the average. It can also be argued that the results for the first few images might be lower than what could have been if the order of presentation was different. Thus the 'learning' concerns might not matter after all the individual results are averaged together.

The subjective quality of the images *lenna* and *motorbike* can be examined from figures 5.12 to 5.17. For each of the two images, the baseline JPEG coded image and the A-JPEG coded images in the two adaptive modes are shown. As explained previously, the overhead mode images are more distorted than their corresponding JPEG compatibility mode counterparts since the same perceptual model parameters are used for all A-JPEG images. It can be seen that the two A-JPEG coded *lenna* images are basically perceptually lossless compared to the baseline JPEG image. The JPEG compatibility mode *motorbike* image is also coded at a perceptually lossless level. Upon close examination at the overhead mode *motorbike* image, some jagginess artifacts can be found on some high activity areas, like the biker's bodies, or the paintings on the helmets or the motorcycle bodies. But the overall perceptual quality of the image is still very close to the baseline JPEG *motorbike* image.

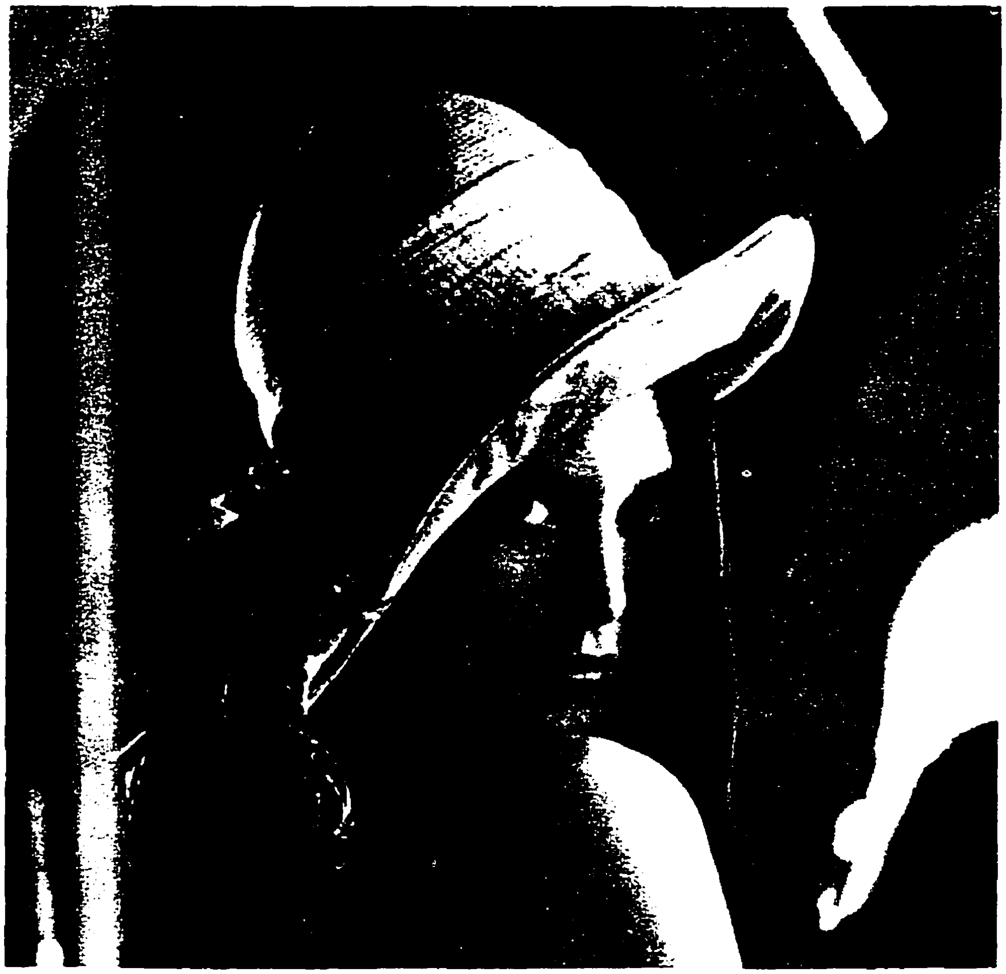


Figure 5.12: The baseline JPEG coded *lenna* image

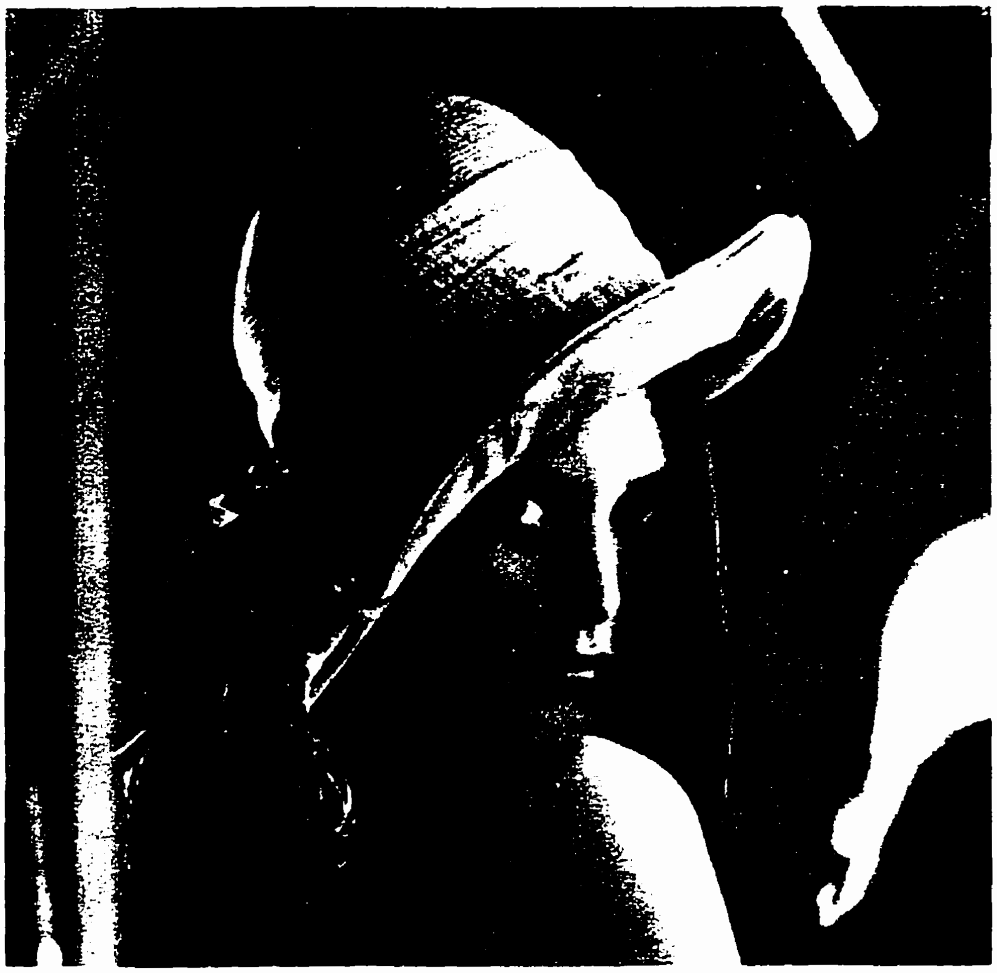


Figure 5.13: The A-JPEG coded *lenna* image - JPEG compatibility mode

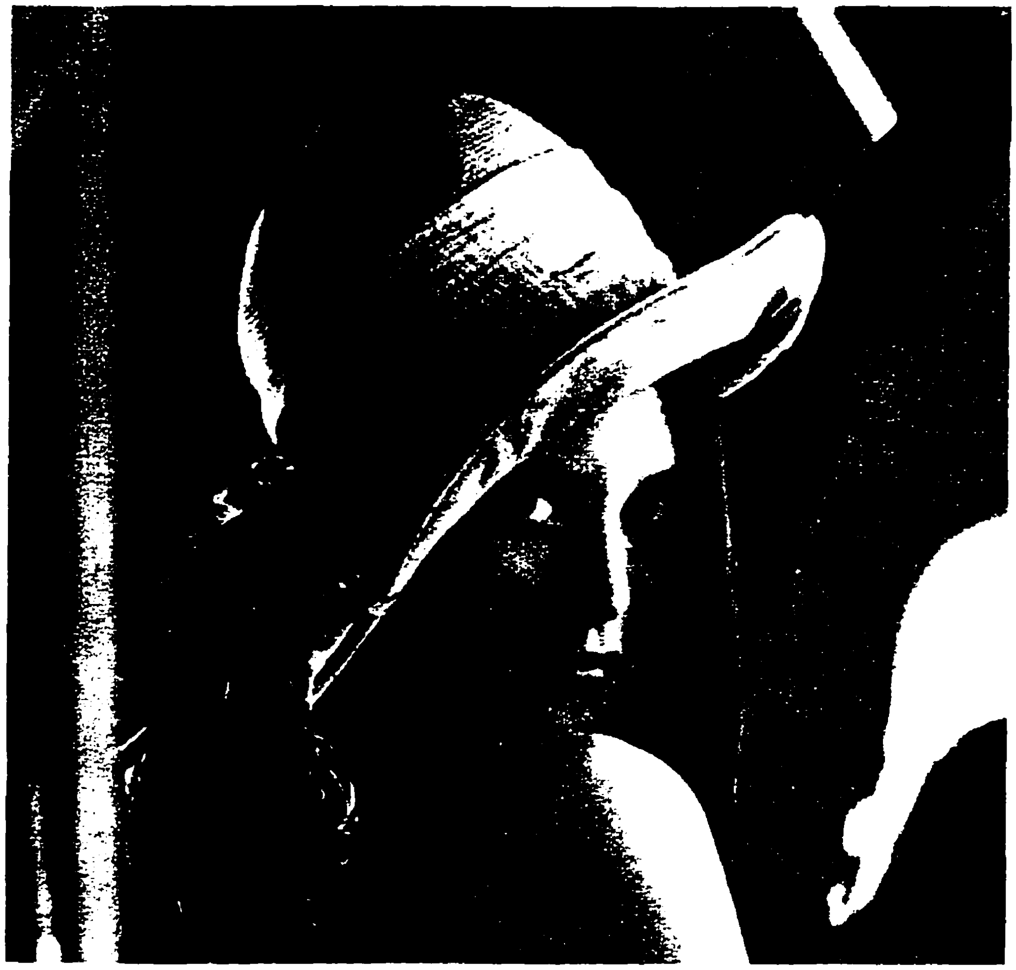


Figure 5.14: The A-JPEG coded *lenna* image - overhead mode

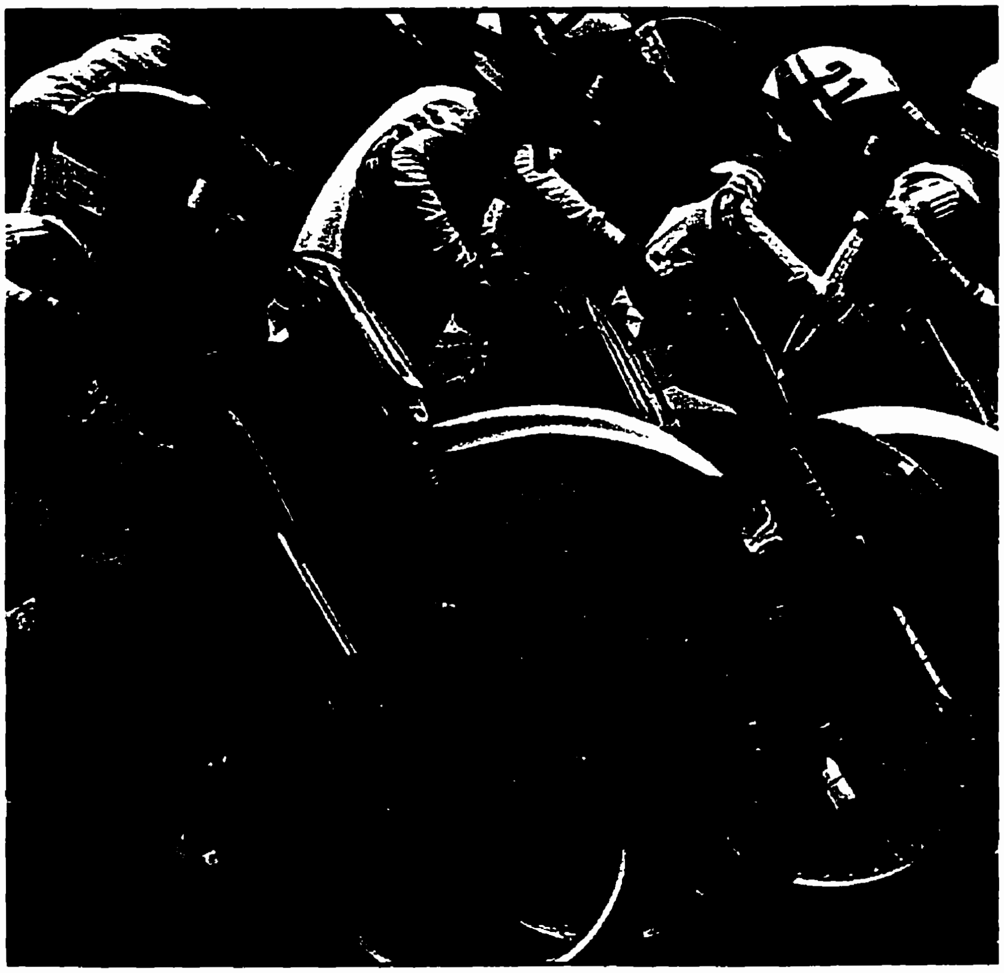


Figure 5.15: The baseline JPEG coded *motorbike* image

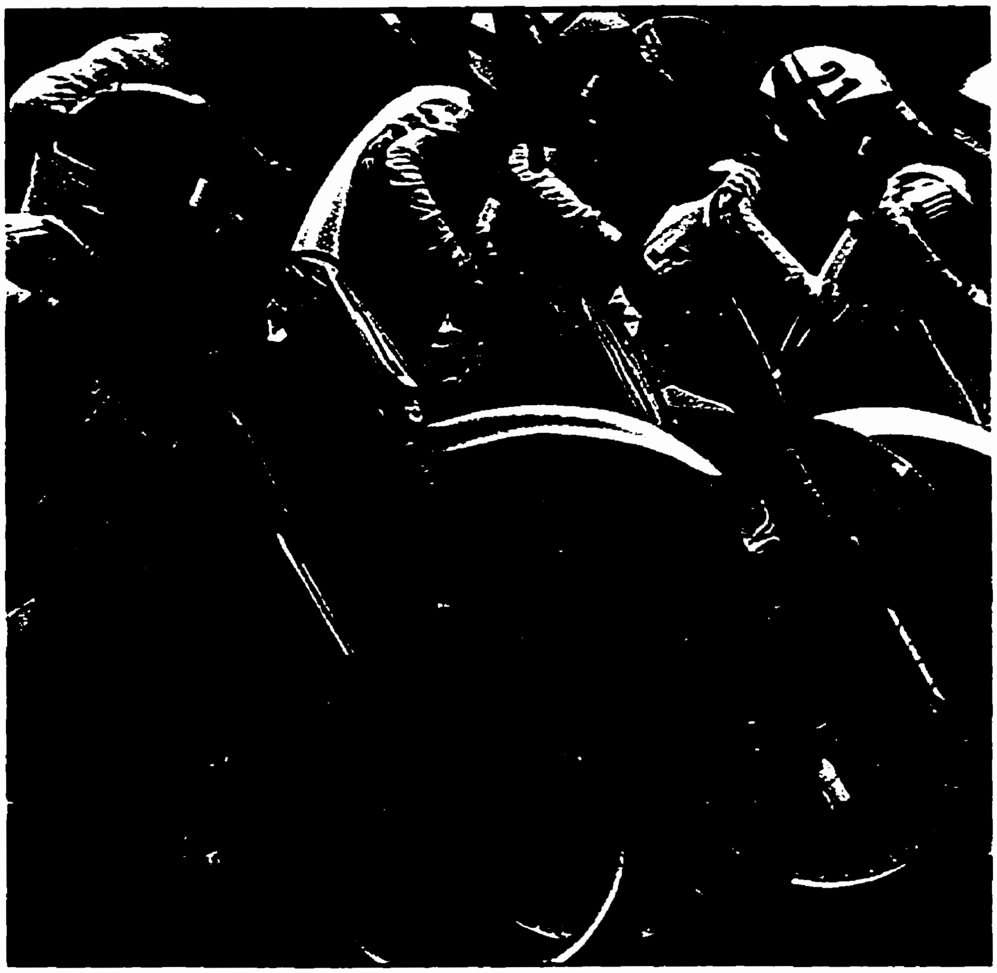


Figure 5.16: The A-JPEG coded *motorbike* image - JPEG compatibility mode

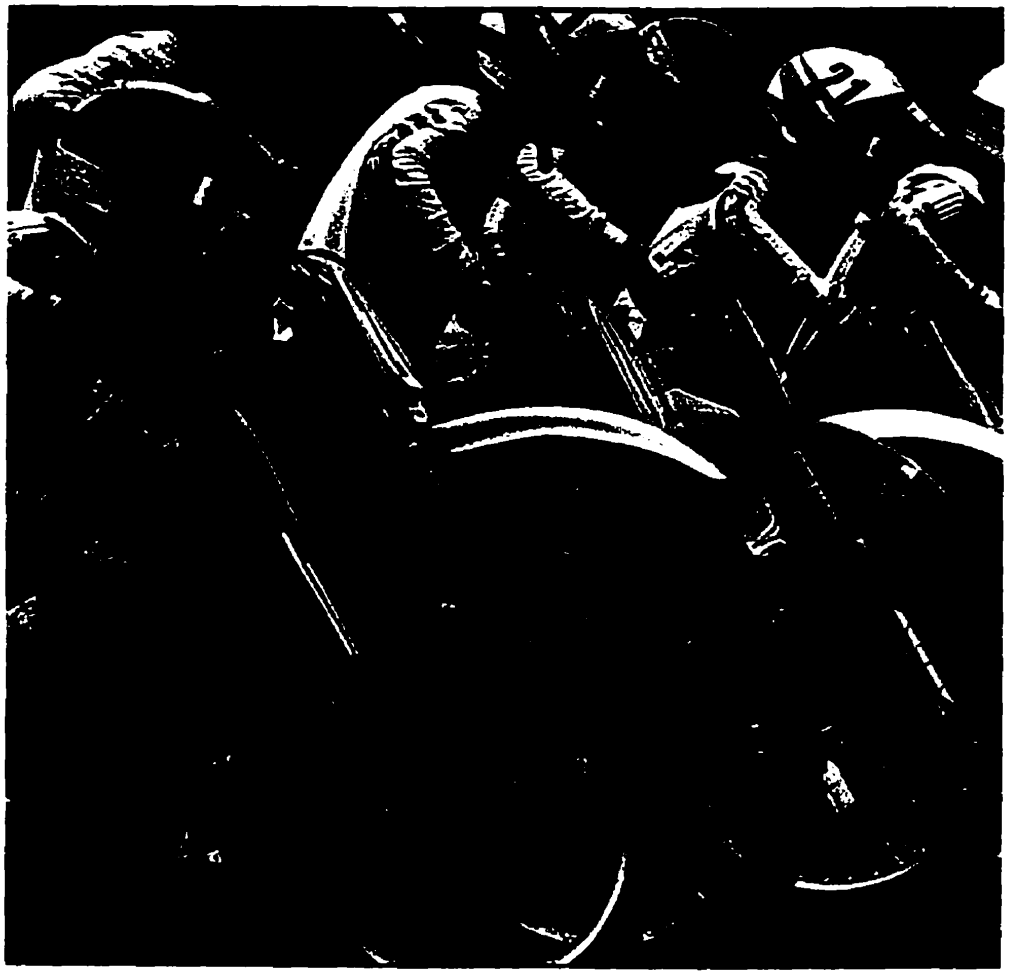


Figure 5.17: The A-JPEG coded *motorbike* image - overhead mode

5.4 Compression results for varying quality factors

The results presented in the previous sections are all obtained using the coder parameters listed in table 5.1. In particular. the JPEG quality factor of 72 was chosen. The value corresponds to a bit-rate of 0.916 bit/pixel for the grayscale image *lenna*. It is generally recognized that at the compressed bit-rate of about 0.75 to 1.5 bit/pixel, the quality of a decompressed colour image is excellent. and is sufficient for most applications [2]. Thus it is insightful to also investigate the adaptive coder's performance for other quality factors to establish a more thorough understanding of the coder's capabilities or shortcomings.

5.4.1 Objective results

Table 5.10 and 5.11 present some compression results for the range of quality factors from 10 to 90. The colour images *lenna* and *houses*, and a grayscale image *barbara* are used. The image *lenna* represents a mainly smooth. low complexity image, *barbara* represents a moderate complexity image, and *houses* represents a high complexity image. The bit-rate column under each image provides the baseline JPEG bit-rate for the corresponding quality factor, and the savings column presents the bit-rate savings in percentage over baseline JPEG with the adaptive coder. Table 5.10 contains the results for the adaptive coder in the JPEG compatibility mode, while table 5.11 is for the overhead mode, with the overhead information LZW coded.

It can be observed from table 5.10 that for the JPEG compatibility mode, the bit-rate savings for a particular image increase with the increase in compression (lower bit-rate). In fact from the experimental data, the savings in bits actually decrease with the decrease in the quality factor. But because the corresponding decrease in bit-rate is even higher, the overall bit-rate savings in percentage still increase.

The results for the overhead mode is quite different from that of the JPEG compatibility mode. The main difference is that the cost of the multiplier map overhead needs to be taken into consideration. The cost of the overhead is a constant for each image, and is independent of the coding bit-rate. Thus the lower the bit-rate, the

Quality factor	lenna(c)		barbara		houses	
	bit-rate (bit/pixel)	savings (%)	bit-rate (bit/pixel)	savings (%)	bit-rate (bit/pixel)	savings (%)
90	2.07	4.4%	2.25	5.3%	3.20	7.1%
80	1.32	4.8%	1.56	6.2%	2.23	9.0%
70	1.01	5.1%	1.26	7%	1.79	10.1%
60	0.83	5.3%	1.07	8.1%	1.51	10.8%
50	0.72	5.6%	0.94	9.5%	1.32	11.4%
40	0.61	5.8%	0.82	10.7%	1.15	12.6%
30	0.50	6%	0.68	12.4%	0.96	13.7%
20	0.38	6.7%	0.51	15.4%	0.72	14.7%
10	0.23	8%	0.30	15.8%	0.43	18.1%

Table 5.10: Bit-rate savings (JPEG mode) for varying quality factors

Quality factor	lenna(c)		barbara		houses	
	bit-rate (bit/pixel)	savings (%)	bit-rate (bit/pixel)	savings (%)	bit-rate (bit/pixel)	savings (%)
90	2.07	6.5%	2.25	11.1%	3.20	14.7%
80	1.32	6.3%	1.56	11.6%	2.23	16.2%
70	1.01	5.6%	1.26	11.1%	1.79	16.7%
60	0.83	4.8%	1.07	11.6%	1.51	16.7%
50	0.72	4.5%	0.94	12.4%	1.32	16.9%
40	0.61	4.2%	0.82	12.5%	1.15	17.5%
30	0.50	2.8%	0.68	12.5%	0.96	17.5%
20	0.38	1%	0.51	12.1%	0.72	16.0%
10	0.23	-4.4%	0.30	3.8%	0.43	12.5%

Table 5.11: Bit-rate savings (Overhead mode - LZW) for varying quality factors

higher the relative cost of the overhead, which offset the potential bit-rate savings. For high baseline JPEG bit-rates. the overhead cost is relatively insignificant and the full adaptive quantization performed in the overhead mode delivers much better performance than adaptive thresholding in the JPEG compatibility mode. However, for lower bit-rates, the relative cost of the overhead starts to hamper the final bit-rate savings result.

It can be observed from table 5.11 that for bit-rates above approximately 0.5 bit/pixel, the bit-rate savings are generally quite stable for all three images. Although apparently *lenna*'s performance is worst since A-JPEG does not perform well for smooth images in general. The bit-rate savings for *barbara* and *houses* are remarkably stable above the baseline JPEG bit-rate of 0.5 bit/pixel. This shows that the raw bit-rate savings achieved by full adaptive quantization are 'just' offset by the overhead costs. However, at bit-rates below 0.5 bit/pixel, it can be observed that there are sharp drop-offs in the final bit-rate savings since the overhead information begins to dominate the final bit-rates. There has been interest in reducing the overhead cost in low-bit-rate situations and [54] presents an algorithm which optimizes the overhead cost for video coding.

5.4.2 Subjective results

Informal comparisons between the baseline JPEG images and the A-JPEG images compressed at the same quality factors by the author show that the two types of images are perceptually very close to each other at the different quality levels examined. In fact the quality factors from 70 to 90 produce images that are so close to the originals that there is no real incentives in using the quality factor 90, which requires a much higher bit-rate than the factor 80 and below.

For low quality compression at bit-rates below about 0.5 bit/pixel, the infamous JPEG blocking artifacts are apparent in the reconstructed images. However, this problem is a general weakness of block-based lossy compression systems and the artifacts exist in both baseline JPEG images and A-JPEG images. The adaptive JPEG coder, designed for high quality compression, is not an effective tool to deal with the blocking artifacts with low-bit-rate compression and there are more established methods that are available [60]. Nevertheless, it is insightful to investigate if the blocking artifacts can be reduced by using the adaptive JPEG coder.

Instead of comparing images compressed using the same quality factor, an informal comparative test was carried out to compare the quality of images compressed at the *same* bit-rates. The rationale is that, if compared to baseline JPEG, A-JPEG requires lower bit-rates at approximately the same perceptual quality for the same quality factor, an A-JPEG image should have 'better' quality than a baseline JPEG image compressed at the same bit-rate. For the image *lenna*, the A-JPEG image with the quality factor (Q) equals 22 has approximately the same bit-rate as a baseline JPEG image with Q = 20. For the image *barbara*, the A-JPEG image with Q = 25 has approximately the same bit-rate as a baseline JPEG image with Q = 20. The JPEG compatibility mode was used so that the multiplier map overhead was not a factor. It was discovered that both of the A-JPEG images have fewer blocking artifacts than their baseline JPEG counterparts, especially in the facial areas of the female subjects in the two pictures. The main reason behind this is that the facial areas are generally classified as plain blocks by block classification in the perceptual model (assume texture masking is the dominant factor for now). Thus the local multipliers in those areas are generally one and they are quantized by the global QM at Q = 22 or 25. The A-JPEG coder can compensate for the higher bit-rates in the plain blocks by more coarsely quantizing the texture blocks with a scaled QM using a lower quality factor at, say, Q = 18. This is the main advantage of adaptive coding, where more bits are used to code the perceptually more important areas. The advantage is also true at higher bit-rates, but it is more apparent at low bit-rate situations, where the artifacts are more visible.

Figure 5.18 shows the graph of the JPEG quality factor as a function of the bit-rate of the compressed image. The results for the three images *lenna, barbara*, and *houses* are presented. The solid lines represent the results for baseline JPEG compression, while the dotted lines represent the results for A-JPEG compression in the JPEG compatibility mode. It can be observed that to achieve the same bit-

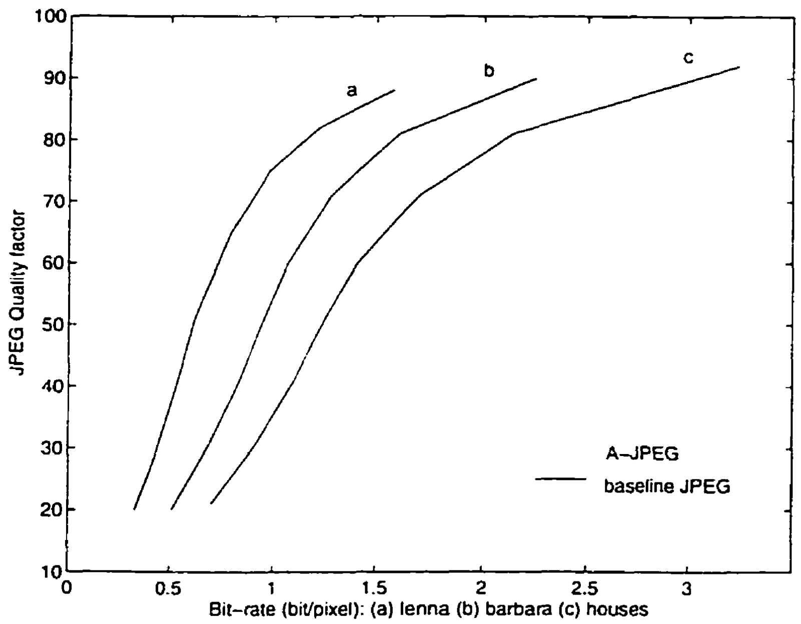


Figure 5.18: The JPEG quality factor as a function of the compressed bit-rate

rate, the solid line (baseline JPEG) always requires a lower quality factor than the corresponding dotted line (A-JPEG) for the same image. In general, the use of a higher quality factor by A-JPEG does not necessarily mean that the A-JPEG image will be perceived 'better' than the baseline JPEG image since the rating of image quality is a highly subjective matter and a reliable perceptual metric that can be computed objectively is still not available. But it can be claimed that the adaptive JPEG coder is able to preserve the quality of the perceptually more important areas such as the plain or the middle-luminance areas, better than baseline JPEG at the same bit-rate by more aggressively compressing image areas that are perceptually less important. In fact, for a high complexity image like *houses* for which A-JPEG performs well, the A-JPEG quality factor can be as many as 9 units higher than that of baseline JPEG at the same bit-rate (50 vs. 41 at 1.1 bit/pixel) .

Figures 5.19 and 5.20 show the baseline JPEG and A-JPEG (JPEG compatibility mode) *barbara* images respectively. Both are coded at 0.51 bit/pixel (Q = 20 for baseline JPEG, Q = 25 for A-JPEG). The facial areas of the images are zoomed in to emphasize any potential difference. The differences are more apparent on a high quality computer monitor, but it can still be observed that A-JPEG provides a slightly better reproduction of *barbara*'s face and the flat area on the left hand side of the images. However, the texture area of *barbara*'s clothes near the lower left hand side is better preserved in the baseline JPEG image, although in fact both images look quite bad in those areas anyway as the blocking artifacts dominate the viewer's attention and texture masking no longer holds at this low bit-rate.

5.5 Computational complexity of the coder

The computational complexity of the coder is examined in this section. One important advantage of the architecture of the adaptive coder is that the computational overhead of the perceptual model is incurred at the encoder level only. The perceptual model only needs to be implemented in a JPEG encoder and the output can be read by any existing standard-compliant JPEG decoder without any overhead cost. Thus the adaptive coder is ideally matched for broadcast-type multimedia applications wheremost of the information is created only once but accessed many times afterwards. In these applications, the computational cost of the encoding overhead of the perceptual model, in light of the overall system usage cost, will be minimal because encoding is done only relatively sparingly. Appendix A provides a more in-depth discussion on the applications of the adaptive coder.

The emphasis of this section is on analyzing the computational overhead of the perceptual model over the total encoding cost of baseline JPEG. The results are presented in table 5.12. The performance is in second, for encoding of the 512 × 512 colour image *lenna*. The performance for baseline JPEG, A-JPEG in the JPEG compatibility mode, and A-JPEG in the overhead mode (multiplier map LZW coded) for different computer processors is presented. The corresponding performance overheads over baseline JPEG in percentage are also calculated for the two adaptive coding modes.

It can be observed that the overheads for the JPEG compatibility mode range from 9% to 14%, while those for the overhead mode range from 6% to 10%. The computational costs mainly come from the calculations in the perceptual model, some

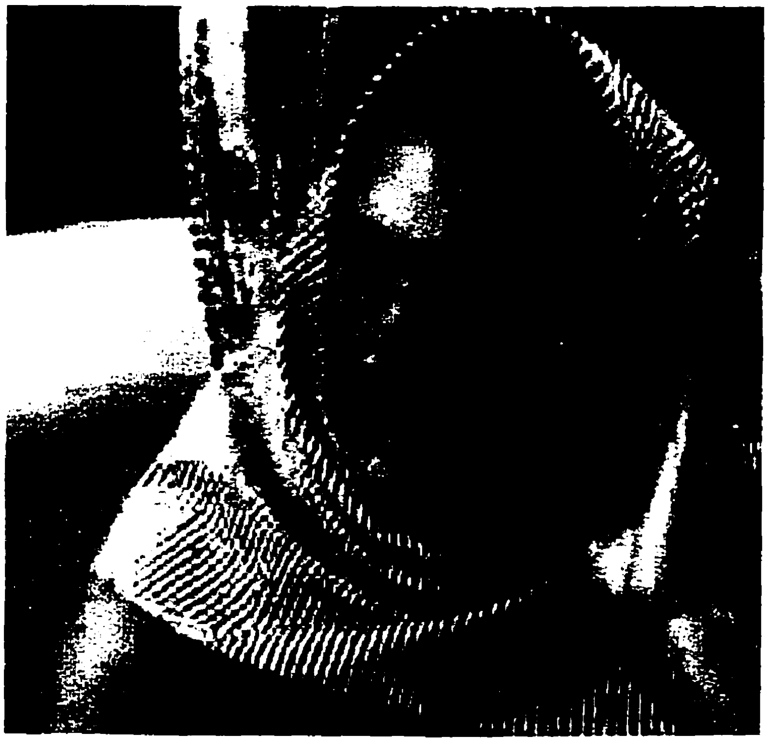


Figure 5.19: The baseline JPEG *barbara* image coded at 0.51 bit/pixel

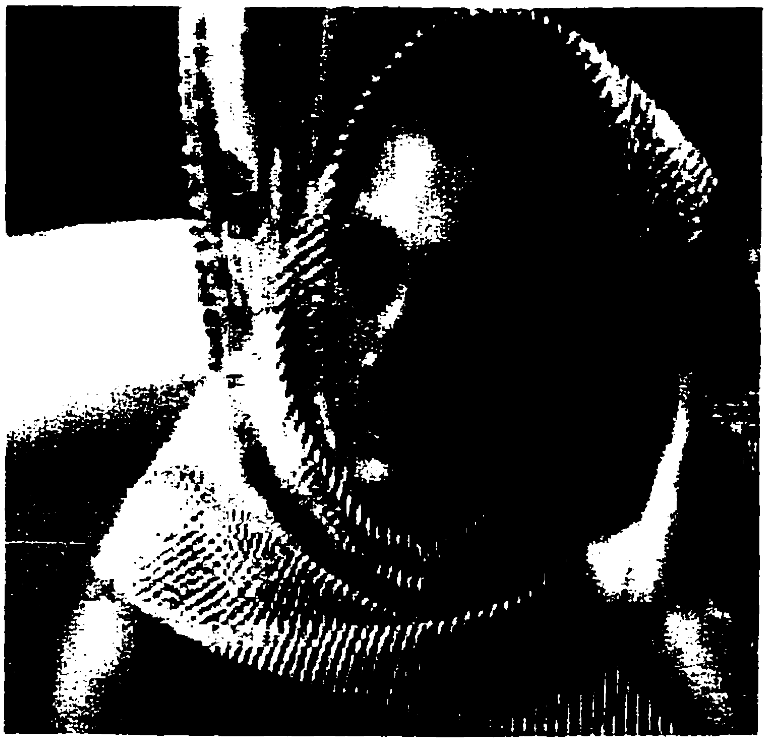


Figure 5.20: The A-JPEG *barbara* image coded at 0.51 bit/pixel (JPEG mode)

Machine type	Baseline JPEG time	Adaptive JPEG			
		JPEG compatibility		Overhead mode	
		time	overhead	time	overhead
Pentium 133	0.79s	0.90s	13.9%	0.87s	10.1%
Pentium Pro 200	0.33s	0.36s	9.1%	0.35s	6.1%
Sun Sparc-5	1.12s	1.23s	9.8%	1.20s	7.1%
Sun Ultrasparc-1	0.49s	0.55s	12%	0.53s	8%

Table 5.12: The A-JPEG encoding computational overhead over baseline JPEG

of the most important operations are:

- Pre-computation of the scaled quantization matrices (QMs) before the parsing of the image blocks so that during quantization the scaled QMs can be called upon quickly by LOT(Look-Up Tables).
- The summations of individual DCT coefficients for the symbols L (low frequency), E (edge), H (high frequency) for block classification.
- Performing block classification using the algorithm presented in section 4.2.2.1.
- The calculation of the mean luminance of the image for luminance masking.

The overhead mode is faster than the JPEG compatibility mode since the JPEG compatibility mode requires an extra adaptive thresholding step before the actual quantization step.

It is expected that with better hardware and software optimization, the encoding computational overhead can be lowered considerably. For instance, the perceptual model uses a lot of repetitive loops to perform summations in block classification and luminance masking. The Intel MMX extensions to the x86 architecture allows up to eight integer pairs to be added together in parallel in one instruction [61] and thus should provide substantial performance improvement for future Intel-based applications.

Overhead for mean luminance calculation

The impact of the mean luminance (mean DC) calculation for luminance masking is substantial since for a 512×512 colour RGB image with 3 channels, $512 \times 512 \times 3 = 786432$ additions are needed. To examine the impact, the sub-routine for the mean luminance calculation was taken out and the adaptive coder was re-run. For the Sparc-5 workstation, the new overhead costs were 7% and 4.5% for the JPEG compatibility mode and overhead mode respectively. These represent improvements of about 30% as compared to the results in table 5.12. Thus a convenient way to speed-up the A-JPEG coder is to fix the mean luminance value at, for example, 128. But this is done at the expense of decreased adaptivity in the luminance masking model and would not be recommended for still image compression in general. However, for video coding, there is generally high correlation between adjacent image frames until a scene change occurs. Thus the perceptual model can take advantage of the previous frame's mean luminance value to reduce the computational cost in a video coding system.

For colour images, another possibility is to calculate the mean luminance from the luminance (Y) component only, after the colour space conversion. This requires a full colour space conversion of the whole image before any DCT and quantization is performed. The current implementation is not able to do that because the coder structure is to perform the whole compression process, including colour space conversion and chrominance subsampling, one horizontal row of 8×8 blocks at a time. With a slight change in the coder structure so that the colour space conversion step for the whole image is completed before the core JPEG coder is invoked, the mean luminance value can be calculated from the luminance channel only and the total number of additions is $512 \times 512 = 262144$, a saving of two third of the cost of the mean luminance calculation as implemented currently.

Chapter 6

Conclusions

A perceptually adaptive DCT-based transform coder (A-JPEG) is implemented in this project. The implementation involves the design of a perceptual model and the implementation of an adaptive coding structure on top of a standard (non-adaptive) JPEG coder implementation. The main objective is to devise a more *perceptually uniform* quantization strategy so that fewer bits are used to represent the perceptually less important areas of an image. The result will be a saving in bit-rate, with no overall loss in perceptual quality.

The perceptual model

The perceptual model contains a texture masking model and a luminance masking model. The key to the texture masking model is a block classification step that differentiates between the plain, edge, and texture blocks so that the local multipliers can be calculated separately for each type of blocks. A new classification algorithm is proposed, which makes use of the ratios between the high frequency and low frequency DCT coefficients of each block to perform the classification. An innovative adaptive luminance masking scheme is also proposed. The scheme is motivated by the observation that in general a bright image needs a different luminance masking strategy than a dark image. The proposed luminance masking scheme adaptively adjusts the luminance masking strategy for an image depending on the image's mean luminance value. A convenient feature of the proposed perceptual model is that both masking models are *linear threshold elevation models* [35] and the user can easily adjust the extent of the distortion introduced by each of the masking models by modifying the value of the maximum elevation factor of the corresponding model.

The two adaptive coding modes

The perceptual model is designed to produce a multiplier map that contains 5-bit scalar multipliers that can be used to scale the quantization step sizes of each individual image block. Two adaptive coding modes are examined in this thesis. In one mode, the multipliers are used for scaling the global quantization matrix during quantization and the multiplier parameters are included with the compressed image data as overhead information for decoding. This adaptive coding mode is referred to as the *overhead mode* in the thesis.

Baseline JPEG compatibility requires a different approach since the original JPEG standard only allows global quantization step sizes. The coder adaptively thresholds some DCT coefficients to zero based on information from the perceptual model. This provides an encoded image output that does not need a multiplier map overhead for decoding and is baseline JPEG compliant. This adaptive coding mode is referred to as the *JPEG compatibility mode.*

Performance - high bit-rate

The performance of the two adaptive coding modes for high quality image compression is examined. In this project, the quality level at which the compressed *lenna* image is about perceptually lossless from the original is used. Using the same JPEG quality factor and perceptual model parameters (table 5.1), for the test image set the bit-rate savings for the JPEG compatibility mode (no overhead) over baseline JPEG range from 3% to 13%, and for the overhead mode, the savings range from 5% to 22%. A subjective test that involves 19 human subjects demonstrates that the A-JPEG coded images are essentially indistinguishable from the baseline JPEG coded images.

The experimental results also show that the adaptive coder provides better performance for images that are hard to compress under baseline JPEG. The reason for

A-JPEG's better performance is that the hard-to-compress areas are generally highly textured, or complex areas that can tolerate more distortion perceptually. Thus A-JPEG is able to identify a lot of those areas in images that are complex and achieve a much better compression.

The results for the overhead mode always surpass those for the JPEG compatibility mode at high bit-rates since the availability of the multiplier map overhead allows more aggressive quantization of the DCT coefficients. However the advantage of the overhead mode over the JPEG compatibility mode is much less for smooth images since A-JPEG does not perform very well for smooth images in general.

Performance - low and medium bit-rate

The perceptual model is designed with high quality image compression as the target application since the JPEG standard is not a very effective tool for low bit-rate, or low quality compression. The texture masking and luminance masking properties of the HVS are not relevant at low bit-rates any more because the blocking artifacts introduced by JPEG will dominate the viewer's attention. Nevertheless, it is insightful to also investigate the performance of A-JPEG at different quality levels since there is a certain gap between the perceptually lossless quality level and the quality level at which the blocking artifacts become annoying. And a user might want to use A-JPEG for a quality level somewhere in between the two extremes.

Section 5.4 shows that the performance of A-JPEG in the JPEG compatibility mode improves with the decrease in bit-rate. But for the overhead mode, the results are different because of the quality-independent multiplier map overhead that also has to be included with the compressed image. The results show that the performance of A-JPEG in the overhead mode is in general quite stable over a wide range of bit-rates. But the resulting bit-rate savings drop off considerably at bit-rates below 0.5 bit/pixel as the overhead information begins to dominate the total bit costs.

Performance - summary

In general. the performance of the A-JPEG coder is affected by two main factors: the source image complexity and the target resulting bit-rate. The project confirms the finding in [31] that the more complex the source image, the better the A-JPEG coder performs. The overhead mode provides substantial improvement over the JPEG compatibility mode for medium to high complexity images. But it is only marginally better for smooth. or low complexity images.

The target bit-rate can affect the choice of the adaptive coding mode. At low bit-rates, say below 0.5 bit/pixel. the performance of the overhead mode is poor since the overhead costs dominate the final bit-rate. The JPEG compatibility mode provides good results for all bit-rates. in fact. the performance of the JPEG compatibility mode improves with the decrease in the resulting bit-rate.

Figure 6.1 summarizes the adaptive coder's performance as a function of the source image complexity and the target baseline JPEG bit-rate. It is only a crude measure but it can serve as a quick reference for determining if the adaptive coder is suitable for compression of a certain kind of image, at a certain bit-rate. The performance of the adaptive coder in the overhead mode can also be directly related to that of the new JPEG extension (section 4.1.1.2). Thus figure 6.1 will also help a JPEG user decide if it is worth switching to the new JPEG extension at the expense of the loss of baseline JPEG decoder compatibility.

Image complexity / target bit-rate	low	medium to high
low	JPEG mode: Good Overhead mode: Poor	JPEG mode: Very good Overhead mode: Fair
medium to high	JPEG mode: Good Overhead mode: Good	JPEG mode: Good Overhead mode: Very good

Figure 6.1: A summary of the performance of the adaptive coder

The scales and terminologies in figure 6.1 are defined quite arbitrarily. But they approximately reflect the previous results for varying quality factors presented in tables 5.10 and 5.11. In figure 6.1, the threshold between low bit-rate and medium to high bit-rate is defined to be 0.5 bit/pixel. The complexity of an image can be approximated by its corresponding baseline JPEG bit-rate. For a fixed quality factor, the higher the resulting baseline JPEG bit-rate, the more complex the image. Thus using the quality factor of 72, which is approximately equivalent to encoding with the JPEG example quantization matrix scaled by a factor of 0.56, a threshold bit-rate of 1.2 bit/pixel is chosen to differentiate between the low complexity and the medium to high complexity images.

For instance, the image *lenna* is a typical low complexity image, *barbara* is a medium complexity image, and *houses* is a high complexity image. The two threshold bit-rates can be applied for both grayscale and colour images since the chrominance channels normally only account for a small percentage of the total bit-rate of a colour image [1]. As for the performance measures, 'very good' refers to a bit-rate saving of above 10%, 'good' refers to a saving of about 4% to 10%, 'poor' refers to a saving of less than 4%, and 'fair' represents a non-consistent saving, somewhere between poor and good.

Computational complexity

The computational complexity of the adaptive coder is modest. Experimentation with different microprocessors shows that for encoding, the computational overhead over baseline JPEG is about 11% for the JPEG compatibility mode, and 8% for the overhead mode. It is expected that with better hardware and/or software optimization, the encoding overhead will go down substantially.

There is no overhead cost for decoding since the perceptual model is only used in the encoder. Thus the adaptive coder is ideally suited for broadcast-type multimedia applications where most of the information is created only once but accessed many times afterwards. Appendix A provides a more general discussion on the applications of the adaptive coder.

6.1 Future work

A lot of work can still be done for further improvement. The following summarizes some of the possibilities for further study:

1. The subjective test in chapter 5 verifies that the human eye is not good at differentiating the small differences between images. A perceptually-motivated objective metric that can be computed (like the common signal-to-noise ratio) is highly preferable to a possibly error-prone, and labour-intensive subjective test. Unfortunately a perceptual metric requires the use of a perceptual model and as of now a universally agreed-upon perceptual model is still not available (otherwise the design of a perceptual model in this project will be deemed unnecessary). A possible choice is a metric that is based on the *just-noticeable-distortion* profile of an image. described in [1, 20]. Some other choices can be found in [59, 62].

2. The perceptual model's two masking elevation parameters as described in table 5.1 are fine-tuned to provide a good trade-off between perceptual quality and coder performance for the test images. However the optimization of perceptual fidelity is a rather subjective problem, and for many individual images, the parameters can still be increased without introducing perceptual loss according to most observers. One of the main reasons behind the need of a perceptual metric is for more robust calibration of the perceptual parameters.

3. The use of the same perceptual parameters for the two adaptive coding modes allows direct comparisons between the performance of the two modes. However, by definition, the overhead mode introduces more distortions to the images than the JPEG compatibility mode (see sections 4.1.1.3). Thus the coder parameters that are designed for good quality in both modes actually under-compress the image in the JPEG compatibility mode. Due to time limitations, a formal analysis has not been performed, but it would be interesting to investigate the JPEG compatibility mode's performance separately.

In an informal test, with an educated guess, the perceptual model's masking elevators are raised so that the JPEG compatibility mode produces an output that has the same size as the output of the overhead mode using the original masking elevators. The two reconstructed images are then compared and there is virtually no perceptual difference between the two, even after close examination with zoom-ins.

4. Frequency sensitivity, one of the HVS properties introduced in chapter 2, has not been investigated in this project. The JPEG quantization matrix (QM) models the HVS's frequency sensitivity. Thus any investigation into frequency sensitivity requires the modification of the QM. In general, QM modification is a global procedure and the analysis is expected to be quite different from the work done in this project since the other HVS properties considered in this project are block level, local masking properties. Some well-known literatures on QM design are in [48, 25].

5. The block classification procedure can also be improved [63]. The classification algorithm's ability to extract edge information is less than perfect since the analysis is done using the DCT coefficients of non-overlapping 8×8 blocks. So the result's precision is much less than that of other classical edge detection techniques, such as the Sobel operators, which use 3×3 kernels for each individual pixel [10] and the current algorithm might miss an edge that happens to locate just near the boundaries of two adjacent blocks.

   In situations where edge features preservation is highly desirable, it might be advantageous to simply turn off the locally adaptive classification correction scheme described in section 4.2.2.1. This ensures that edge blocks that are close to texture blocks will not be re-classified as texture blocks. With some slight change in the design, the user might also add a preprocessing module to perform a separate edge detection analysis. The results can then be used to aid the block classification decision. With the fast computer processors currently available, the added overhead will be small[1] and may not be a big performance bottleneck to overcome for non-realtime use. Furthermore, to take advantage of the inherently varying block statistics of real-life images, a fuzzy classification method might also be used [64].

6. For colour images, only the luminance properties of the HVS are considered in the perceptual model. Reference [14] introduces some issues concerning the masking properties of the chrominance channels that can be very useful for further improvement of the perceptual model.

   There has also been report that the use of a perceptually uniform colour space like the CIE[2] L*a*b* and L*u*v* colour spaces provides better JPEG compression performance than the traditional RGB and YIQ (similar to the YCrCb colour space used in JPEG) colour spaces [18]. As discussed in chapter 3, the colour space conversion module is independent of the core JPEG coder. Thus the change in colour space does not require major coder modification[3] and it would be interesting to investigate if the A-JPEG coder can also benefit from the use of a new colour space.

7. The perceptual model proposed in this project can also be implemented in a MPEG (Moving Picture Expert Group) [32] video coding system since MPEG is also based on the DCT-based block transform coding model and MPEG's intra-coding mode is basically identical to JPEG image coding. However video coding involves substantially more design issues than image coding and the usefulness of the perceptual model in a video coding environment is still to be investigated. A comparison between the results obtained with MPEG's standard adaptive quantization scheme [65] and the results obtained with the proposed perceptual model is to be examined [37, 44, 45]. Moreover, in video coding, there is also

[1]Performing the edge detection function with the popular *xv* program takes less than one second in the Linux OS on a Pentium Pro 200 PC.

[2]Commission Internationale de L'Eclairage.

[3]The perceptual model requires the luminance channel Y. Thus the new colour space conversion module may need to compute an additional Y channel for interim use by the perceptual model during encoding.

the added opportunity for *temporal masking* [14] that can be implemented in the perceptual model for further improved results.

Appendix A

Possible applications for adaptive JPEG coding

This appendix attempts to provide and discuss about some application scenarios in which the adaptive coder might prove useful. An important characteristic of the adaptive coder is that all the perceptual overhead are done in the encoder only. The decoding performance for an A-JPEG compressed file is the same as that for a baseline JPEG file. Thus any *decoding-heavy* applications will benefit from a smaller A-JPEG file size, while at the same time the cost of the encoding overhead, in light of the overall system usage cost, will be minimal because encoding is done sparingly.

Asymmetric multimedia communications

One application scenario is in multimedia communications. With the continuous advancement of computer processing power and display device technology, and the rapidly rising popularity of the Internet, multimedia information, in particular, image and video[1] data, are now very much within reach for even casual computer users. One

[1]The perceptual model's usefulness in a video coding scheme like MPEG has not been tested. Nevertheless, in some applications, Motion-JPEG (application of JPEG on a sequence of pictures, disregarding temporal redundancy - there is no recognized standard for Motion-JPEG, thus it is mostly a proprietary format) has also been used in video applications, e.g. the Míro miroVIDEO DC30 digital video editing system uses Motion-JPEG for studio-quality video compression. Thus the discussions on video is not totally irrelevant for A-JPEG.

characteristic of multimedia information is that they are, in most cases, accessed by decoding-heavy applications or systems. In other words, multimedia information resembles *Write-Once, Read-Many (WORM)* information. For instance, the front page images of Cable News Networks' (CNN) world wide web home page[2] are accessed by thousands of hits everyday world wide, but the images are only created once. The same is true for millions of JPEG files on the Internet.

Entering the digital age, the consumer electronics industry and the segment of the computer industry that is concerned with multimedia technology are starting to merge already. And the term *playback* used in consumer electronics can easily be related to *decoding* in computer terminology as decoding of digitized multimedia data (compressed or not) is always needed before the actual reproduction, or playback. Similarly, the multimedia term *encoding* can be loosely regarded to in consumer electronics terminology as *recording digitized data, with or without compression, into a standard format for later reproduction.* Examples of the aforementioned decoding-heavy (*playback-heavy*) applications are also abundant in consumer electronics. For instance, the laserdiscs or movie tapes sold or rented in the local video stores are designed for playback only. A similar case can be made for audio data for the millions of music CDs sold every year worldwide. In fact the broadcasting of television programs to the national households everyday is a perfect example of the *record-once, playback-many* scenario. This link between the computer and consumer electronics industry is important since sooner or later the present analog video signal used by the consumer electronics industry will be replaced by digital signal (possibly in the DVD format) and JPEG and MPEG are expected to play an important role in helping to reduce the transmission and storage costs.

As an adaptive scheme designed for reducing storage costs for image information encoded using the JPEG standard, A-JPEG offers an attractive option for storage of these 'decoding-heavy' image data for a small one-time encoding cost. This promise of A-JPEG also echoes a main advantage of vector quantization and fractal coding, for which the decoding times are much smaller than the encoding times [4, 66]. The

[2]http://www.cnn.com

difference being that for the latter two coding schemes. the encoding overheads are actually several magnitudes higher than that for decoding. This encoding-decoding inequality of multimedia data can also be related to the design philosophy of the Asymmetric Digital Subscriber Line (ADSL) [67], where the download bandwidth is much higher than the upload bandwidth. This reflects a fact that the average consumers have relatively more interest in receiving information (and subsequently decoding it). than creating information (encoding).

Please note that A-JPEG will be useful only when the multimedia data are suitable for the DCT-based transform coding compression method used by JPEG. namely. continuous-tone photographic image data. Two exceptions are computer graphics images. which are used extensively in the video and computer game industry: and when there are needs for near-perfect quality reproduction. like those demanded by big-screen IMAX movies and Kodak Photo CD images [3]. Nevertheless. in most practical situations where real-life images and video are involved. the JPEG and MPEG standards have been proven to be most useful. In fact the Grand Alliance for HDTV has chosen MPEG as the compression scheme for the future digital HDTV standard in North America.

The two groups of prospects

As an example. consider a 1 GByte hard drive for storing image and video data. A 7.5% reduction in the storage requirement will translate into an additional 75 MBytes of disk space. Assume that one high quality colour JPEG image consumes on average 50.000 bytes, the additional disk space will allow the storage of 1,500 more images. Two different groups can be identified as potential customers who might be interested in the A-JPEG technology. The first group consists of the majority of *multimedia content providers*, who will obviously benefit from the reduced storage cost provided by A-JPEG. For instance, a lot of the Internet web page content providers will likely be interested. This includes any company that has a sizable presence on the world wide

[3]Photo CD's further on-line information: http://www.kodak.com

web. Multimedia CD-ROM makers, like those that produce digital encyclopedias, will also be interested because more images can be put into the CD-ROM. Other prospects include any maintainer of large image databases, and future DVD and HDTV software providers (a.k.a. Hollywood studios). The definition of multimedia content providers is not limited to commercial organizations only. Any individual consumer who uses a scanner to digitize a picture has produced multimedia content already. And he or she will benefit from using A-JPEG as the size of the library of scanned pictures increases. Similarly, as digital cameras and digital camcorders start to become more popular, A-JPEG will provide an attractive alternative to reduce storage cost since portable storage media are typically more expensive and have lower capacity than their desktop counterparts.

Besides multimedia content providers, the other group of prospects of A-JPEG includes the software and hardware companies that will be interested in implementing the A-JPEG coder. The main difference between the two groups is that the first group consists of companies or individuals that will be interested in *using* the A-JPEG coder. However, not all parties in this group are necessarily going to implement the coder in-house. This is especially true for individual consumers, who will likely purchase a software coder off-shelf, or use the bundled software that comes with the scanner, digital camera, etc.

Thus this creates a need for the second group of prospects, which contains companies that will be more interested in *implementing* the coder for use by other parties. This group can be further classified into two subgroups, software companies and hardware manufacturers. Examples of *software companies* include companies that produce image processing, image database, and Motion-JPEG/MPEG encoder softwares. A low complexity implementation of the A-JPEG coder actually does not require much research and development expertise, so software firms that specialize in delivering customized software solution for business will likely be interested as well. As for *hardware manufacturers*, firms that produce customized JPEG encoding chips will obviously be interested. Other prospects include companies that manufacture the consumer electronics gadgets such as the digital scanners, or digital cameras discussed in the preceding paragraphs. These firms, however, do not necessarily need to design the encoding chips in-house, they can just purchase the needed hardware components from a third party company. Alternatively, these firms can choose to use software instead by providing bundled softwares that the customers can use once the digitized data are captured. Tables A.1 and A.2 summarize the two groups of prospects who might be interested in using A-JPEG for 'decoding-heavy' multimedia information.

Subgroup Name	Description
Multimedia Content providers (companies)	Web page content providers
	Multimedia CD-ROM makers
	Maintainers of image/video databases
	Digital video-on-demand providers
	Future DVD, HDTV software providers
Individual consumers	Users of digital scanners, digital cameras,
	digital camcorders, and future DVD recorders

Table A.1: Potential *users* of A-JPEG

Subgroup Name	Description
Software companies	Image processing software
	Image database/archival software
	Motion-JPEG/MPEG encoders
	Customized software solution providers
Hardware companies	Makers of customized DSP chips
	Makers of digital scanners, digital cameras,
	digital camcorders, and future DVD recorders

Table A.2: Potential *implementers* of A-JPEG

Symmetric multimedia communications

Although the majority of multimedia information is decoding-heavy, there are cases in which encoding is done just as frequently as decoding. In these situations the encoding overhead of A-JPEG will need to be taken into consideration carefully. Examples of these scenarios are in video conferencing, video phone, or live video broadcasting applications, where Motion-JPEG might be used for video coding. As these are all real-time applications, any encoding or decoding delay will possibly result in quality deterioration. A detailed analysis of the channel bandwidth and cost, encoding/decoding time, quality requirement, and the statistics of the target video sources is needed for a more complete evaluation. The following provides some arguments as to why A-JPEG might still be useful for these applications:

- Buffer control is needed to maintain a constant bit-rate for video conferencing. By reducing frame size for a given quality factor, A-JPEG provides a tradeoff of slightly increased encoding time but better buffer utilization.
- It can be argued that at present the network bandwidth is a bigger limiting factor than the encoder processing speed. Moreover, network characteristics will also affect the overall system performance. The internet and the ethernet-based local area networks commonly encountered nowadays are generally not designed for video delivery. However, as network bandwidth increases, at some point the processing speed might become the limiting factor.
- Hardware acceleration or better software optimization should lower the encoding overhead considerably. More advanced microprocessor technology will also help, such as the Intel MMX extensions to the x86 processor architecture (section 5.5).

Bibliography

[1] N. Jayant, J. Johnston, and R. Safranek. Signal compression based on models of human perception. *Proceedings of the IEEE*, 81(10):1385–1422, Oct 1993.

[2] G. K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):31–44, Apr 1991.

[3] W. B. Pennebaker and J. L. Mitchell. *JPEG still image data compression standard.* Van Nostrand Reinhold, New York, 1993.

[4] V. Bhaskaran and K. Konstantinides. *Image and video compression standards: algorithms and architectures.* Kluwer Academic Publishers, Boston, 1995.

[5] T. M. Cover and J. A. Thomas. *Elements of information theory.* Wiley, New York, 1991.

[6] R. J. Clarke. *Transform coding of images.* Academic Press, London, 1985.

[7] R. J. Clarke. *Digital compression of still images and video.* Academic Press, London, 1995.

[8] J. L. Mannos and D. J. Sakrison. The effects of a visual fidelity criterion on the encoding of images. *IEEE Trans. on Information Theory*, IT-20(4):525–536, Jul 1974.

[9] A. N. Netravali and B. Prasada. Adaptive quantization of picture signals using spatial masking. *Proceedings of the IEEE*, 65(4):536–548, Apr 1977.

[10] R. C. Gonzalez and R. E. Woods. *Digital image processing.* Addison-Wesley. Reading, Massachusetts, 1992.

[11] D. A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings Institute of Electrical and Radio Engineerings*, 40(9):1098–1101, Sep 1952.

[12] I. H. Witten, R. M. Neal, and J. G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520–540. Jun 1987.

[13] The Independent JPEG Group's JPEG software distribution, release 6a, 1996. Internet anonymous ftp location: ftp.uu.net:graphics/jpeg/jpegsrc.v6a.tar.gz.

[14] A. N. Netravali and B. G. Haskell. *Digital pictures: representation. compression, and standards.* Plenum Press, New York, 1995.

[15] R. M. Gray. Vector quantization. *IEEE ASSP Magazine.* 1(2):4–29. Apr 1984.

[16] J. W. Woods and S. D. O'Neil. Subband coding of images. *IEEE Trans. on Acoustics. Speech. and Signal Processing*, ASSP-34(5):1278–1288, Oct 1986.

[17] M. Vetterli and J. Kovacevic. *Wavelets and subband coding.* Prentice-Hall. Englewood Cliffs, New Jersey, 1995.

[18] N. M. Moroney and M. D. Fairchild. Color space selection for JPEG image compression. *Journal of Electronic Imaging*, 4(4):373–381, Oct 1995.

[19] G. Buchsbaum. Visual system considerations in the coding of natural color images. In A. B. Watson, editor, *Digital Images and Human Vision.* MIT Press, Cambridge, MA, 1993.

[20] C. H. Chou and Y. C. Li. A perceptually tuned subband image coder based on the measure of just-noticeable-distortion profile. *IEEE Trans. on Circuits and Systems for Video Technology*, 5(6):467–476, Dec 1995.

[21] J. O. Limb. Distortion criteria of the human viewer. *IEEE Trans. on Systems, Man, and Cybernetics*, SMC-9(12):778–793, Dec 1979.

[22] R. L. de Queiroz and K. R. Rao. Transform coding. In *Handbook of visual communications*. Academic Press, London, 1995.

[23] H. Lin and A. N. Venetsanopoulos. Incorporating human visual system (HVS) models into the fractal image compression. In *Proc. IEEE ICASSP*, pages 1950–1953, 1996.

[24] A. Netravali, E. Petajan, S. Knauer, K. Matthews, R. J. Safranek, and P. Westerink. A high quality digital HDTV codec. *IEEE Trans. on Consumer Electronics*, 37(3):320–330, Aug 1991.

[25] A. C. Hung and T. H. Y. Meng. Optimal quantizer step sizes for transform coders. In *Proc. IEEE ICASSP*. pages 2621–2624, 1991.

[26] M. Crouse and K. Ramchandran. Joint thresholding and quantizer selection for decoder-compatible baseline JPEG. In *Proc. IEEE ICASSP*, pages 2331–2334. 1995.

[27] H. Lohscheller. A subjectively adapted image communication system. *IEEE Trans. on Communications*, COM-32(12):1316–1322, Dec 1984.

[28] R. J. Safranek and J. D. Johnston. A perceptually tuned sub-band image coder with image dependent quantization and post-quantization data compression. In *Proc. IEEE ICASSP*, pages 1945–1948, 1989.

[29] R. J. Safranek. A comparison of the coding efficiency of perceptual models. In *SPIE Conf. on Human Vision, Visual Processing, and Digital Display*, volume 2411, pages 83–91, 1995.

[30] ISO/IEC 10918-3. *Digital compression and coding of continuous-tone still images, part 3, extensions*, working draft edition, 1994.

[31] R. J. Safranek. A JPEG compliant encoder utilizing perceptually based quantization. In *SPIE Conf. on Human Vision, Visual Processing, and Digital Display*, volume 2179, pages 117–126, 1994.

[32] D. LeGall. MPEG: a video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, Apr 1991.

[33] ITU-T Rec. H.261. *Video codec for audiovisual services at px64 kbit/s*, 1993.

[34] ITU-T Rec. H.263. *Video coding for low bit rate communication*, 1996.

[35] T. D. Tran. A locally adaptive perceptual masking threshold model for image coding. Master's thesis, Massachusetts Institute of Technology, 1994.

[36] W. H. Chen and C. H. Smith. Adaptive coding of monochrome and color images. *IEEE Trans. on Communications*, COM-25(11):1285–1292, Nov 1977.

[37] A. Puri and R. Aravind. Motion-compensated video coding with adaptive perceptual quantization. *IEEE Trans. on Circuits and Systems for Video Technology*, 1(4):351–361, Dec 1991.

[38] B. Ramamurthi and A. Gersho. Classified vector quantization of images. *IEEE Trans. on Communications*, COM-34(11):372–382, Nov 1986.

[39] Y. S. Ho and A. Gersho. Classified transform coding of images using vector quantization. In *Proc. IEEE ICASSP*, pages 1890–1893, 1989.

[40] M. H. Lee and G. Crebbin. Classified vector quantisation with variable block-size dct models. *IEE Proc.-Vis. Image Signal Process.*, 141(1):39–48, Feb 1994.

[41] J. W. Kim and S. U. Lee. A transform domain classified vector quantizer for image coding. *IEEE Trans. on Circuits and Systems for Video Technology*, 2(1):3–14, Mar 1992.

[42] K. W. Chun, K. W. Lim, H. D. Cho, and J. B. Ra. An adaptive perceptual quantization algorithm for video coding. *IEEE Trans. on Consumer Electronics*, 39(3):555–558, Aug 1993.

[43] J. Park, J. M. Jo, and J. Jeong. Some adaptive quantizers for HDTV image compression. In L. Stenger, L. Chiariglione, and M. Akgun, editors, *Signal processing of HDTV, V*. Elsevier, New York, 1994.

[44] J. Sun, W. Zhang, and S. Yu. Design of adaptive quantizer for MPEG video coding. In *SPIE Conf. on Advanced Image and Video Communications and Storage Technologies*, volume 2451, pages 320–327, 1995.

[45] A. Sultan and H. A. Latchman. Adaptive quantization scheme for MPEG video coders based on HVS (human visual system). In *SPIE Conf. on Digital Video Compression: Algorithms and Technologies*, volume 2668, pages 181–188, 1996.

[46] S. H. Tan, K. K. Pang, and K. N. Ngan. Classified perceptual coding with adaptive quantization. *IEEE Trans. on Circuits and Systems for Video Technology*, 6(4):375–388, Aug 1996.

[47] R. Forchheimer and T. Kronander. Image coding - from waveforms to animation. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, 37(12):2008–2023, Dec 1989.

[48] A. B. Watson. DCT quantization matrices visually optimized for individual images. In *SPIE Conf. on Human Vision, Visual Processing, and Digital Display IV*, volume 1913, pages 202–216, 1993.

[49] D. F. Shen and Wang S. C. Measurements of JND property of HVS and its applications to image segmentation, coding and requantization. In *SPIE Conf. on Digital Compression Technologies and Systems for Video Communications*. volume 2952, pages 113–121, 1996.

[50] CCIR Rec. 500-2. *Method for the subjective assessment of the quality of television pictures*, 1982.

[51] H. Murakami, H. Hashimoto, and Y. Hatori. Quality of band-compressed TV services. *IEEE Communications Magazine*, 26(10):61–69, Oct 1988.

[52] R. Rosenholtz and A. B. Watson. Perceptual adaptive JPEG coding. In *Proc. IEEE ICIP*, volume I, pages 901–904, 1996.

[53] P. H. S. Truong and S. C. Y. Ho. Block adaptive classified vector quantization. In *SPIE Conf. on Digital Video Compression: Algorithms and Technologies*, volume 2419, pages 340-351. 1995.

[54] A. Ortega and K. Ramchandran. Forward-adaptive quantization with optimal overhead cost for image and video coding with applications to MPEG video coders. In *SPIE Conf. on Digital Video Compression: Algorithms and Technologies*. volume 2419, pages 129–138, 1995.

[55] J. A. Storer, editor. *Image and text compression*. Kluwer Academic Publishers, Boston, 1992.

[56] T. A. Welch. A technique for high-performance data compression. *IEEE Computer*, 17(6):8–19, Jun 1984.

[57] I. Rabinovitch. High quality image compression using the wavelet transform. Master's thesis, University of Toronto, 1996.

[58] D. R. Fuhrmann, J. A. Baro, and J. R. Cox. Experimental evaluation of psychophysical distortion metrics for JPEG-encoded images. In *SPIE Conf. on Human Vision, Visual Processing, and Digital Display IV*, volume 1913, pages 179–190, 1993.

[59] H. Marmolin. Subjective MSE measures. *IEEE Trans. on Systems. Man, and Cybernetics*, SMC-16(3):486–489, May/Jun 1986.

[60] H. S. Malvar and D. H. Staelin. The LOT: transform coding without blocking effects. *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSP-37:553-559, Apr 1989.

[61] Intel Corporation. *MMX[tm] Technology Software Developer FAQs*, 1997. WWW location: http://www.intel.com.

[62] S. Daly. The visible difference predictor: an algorithm for the assessment of image fidelity. In A. B. Watson, editor, *Digital Images and Human Vision*. MIT Press, Cambridge, MA, 1993.

[63] E. Leung. A new perceptually adaptive JPEG. Bachelor's thesis, University of Toronto, 1997.

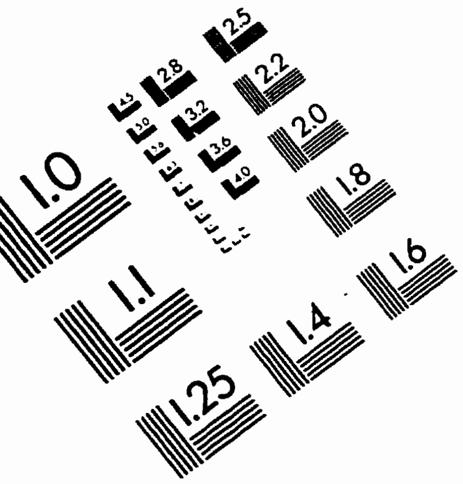
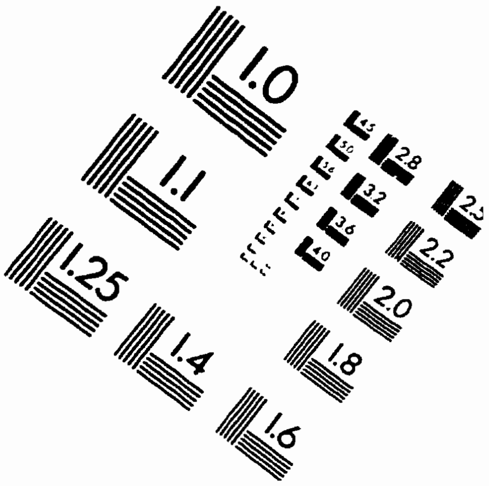
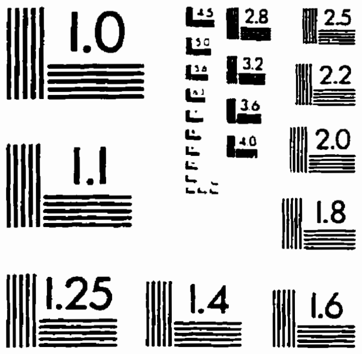
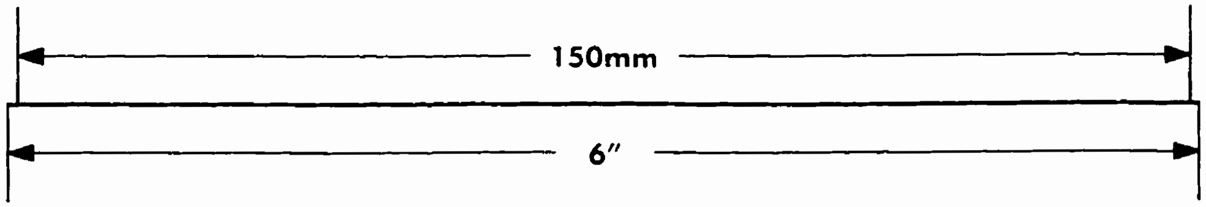
[64] L. Corte-Real and A. P. Alves. A fuzzy classified vector quantizer for image coding. *IEEE Trans. on Communications*, COM-43(2/3/4):207–215, Feb/Mar/Apr 1995.

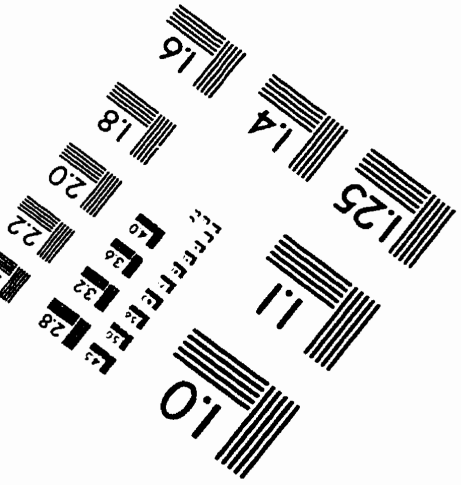
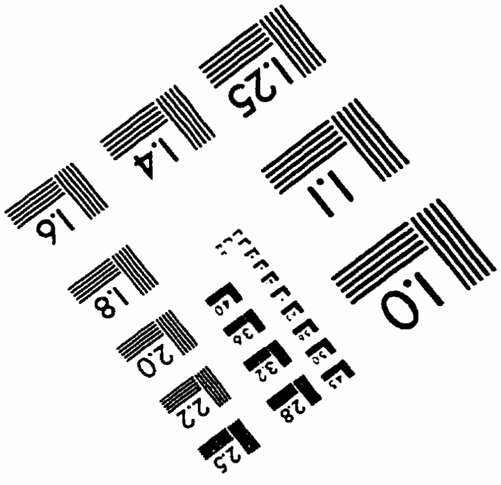
[65] ISO/IEC JTC1/SC29/WG11/N0400. *MPEG-2 Test Model 5*, Apr 1993.

[66] Y. Fisher, editor. *Fractal image compression: theory and application*. Springer-Verlag, New York, 1995.

[67] C. N. Judice. Visual communications in the US. *IEICE Trans. on Communications*, E75-B(5):309–312, May 1992.

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE . Inc
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989