Application of Image Compression Algorithms to Realistic Image Synthesis

Mark Bolin

CIS-TR-95-19
December 1995

Department of Computer and Information Science
University of Oregon

APPLICATION OF IMAGE COMPRESSION ALGORITHMS TO REALISTIC IMAGE SYNTHESIS

by

MARK R. BOLIN

Nov 22, 1995

Faculty Sponsor:

Gary Meyer

Committee Members:

Kent Stevens
Andrzej Proskurowski

A Directed Research Project

Presented to the Department of Computer and Information Science and the Graduate School at the University of Oregon in partial fulfillment of the requirements of the Doctoral degree

Portions of this Directed Research Project were subsequently published in:

Bolin, M. R. and Meyer, G. W., "A Frequency Based Ray Tracer," *Computer Graphics, Annual Conference Series*, ACM SIGGRAPH, pp. 409-418, 1995.

Abstract

An algorithm has been developed that allows realistic images to be directly synthesized into the frequency representation that forms the first stage of the JPEG compression algorithm. This representation allows elements of compression theory to be directly applied to the image synthesis algorithm in order to reduce the amount of sampling necessary to generate a synthetic image. Furthermore, this technique will be shown to offer a novel solution to a number of the difficulties involved in the sampling and reconstruction problem inherent in ray tracing a synthetic scene.

1 Introduction

Image compression algorithms are increasingly being applied to digital images in order to reduce the storage and transmission costs associated with those images within a digital environment. Traditionally, image synthesis algorithms have painstakingly computed every bit and pixel of the image. This process results in a large amount of wasted effort when the image is then converted to a compressed format and a significant portion of the information is discarded as a result of its visual insignificance.

A survey of modern image compression algorithms (see Appendix A) shows that increasing attention is being paid to the visual importance of information contained within an image. Ideally an image compression algorithm is able to arrange the information within an image in terms of the visual significance of that information. This allows the least significant information to be discarded first and increasingly significant information next as greater compression rates are desired.

The JPEG still image compression standard performs such an operation and is the most widely used of modern image compression algorithms. It achieves this ordering of visual significance by converting an image into the frequency domain by means of the

Discrete Cosine Transform. Within this frequency domain the algorithm is then able to take advantage of the low frequency energy prevalent in most images, as well as the greater sensitivity of the human visual system to low frequency information. The compression mechanism is therefore able to assign more bits to the representation of the more important low frequency information and less to the less significant high frequency information.

The image synthesis techniques currently employed by the computer graphics community have yet to adequately exploit the relative importance of image frequency information within their rendering algorithms. This is a result of the spatial domain in which they operate. Image synthesis within the spatial domain additionally limits the ability of an algorithm to detect undersampled or aliased regions within an image. This causes a lack of accuracy within the existing adaptive sampling metrics and results in unnecessary samples being taken. Furthermore, the reconstruction and filtering of samples within the spatial domain is a difficult and expensive process. The limitations of this process have resulted in the introduction of aliasing artifacts and noise into the resulting images.

This report presents an alternative image synthesis technique that iteratively generates images directly into the frequency domain that forms the first stage of the JPEG compression algorithm. This technique allows an image to be generated and refined in a manner that is analogous to the compressed representation of the image at successively higher visual quality levels. This method not only reduces the cost of compressing the resultant image but also drastically reduces the wasted effort involved in generating non-visually significant information that would be discarded by a subsequent compression step. In addition, this technique permits the synthesis algorithm to directly exploit image frequency information during its progression, and provides a mechanism capable of solving many of the difficulties associated with sampling, reconstruction and filtering within the spatial domain.

The order in which the new algorithm generates and refines frequency terms is equivalent to the low to high ordering present within the JPEG algorithm. This allows the algorithm to concentrate first on the most significant low frequency terms and, progressively remove the residual noise from the less significant higher frequency terms as time and resources permit. In addition, this low to high frequency refinement of the image is capable of providing the user with early feedback and provides for an excellent early and intermediate representation of the image.

A significant feature of the Discrete Cosine Transform is its ability to compact the majority of the energy present within an image into the first few frequency terms. This compaction is especially advantageous for the image synthesis technique presented within this report, in that it implies that the majority of the regions within the image will be able to be represented with the first few frequency terms. This permits the synthesis of most regions of the image to be quickly computed while only a small portion of the image will require the greater sampling necessary to determine high frequency content.

Additionally, the iterative frequency representation generated by the algorithm provides a novel means of detecting which areas of the image currently contain the most objectionable artifacts. This allows the sampling of the scene to be adaptively controlled such that these areas will be refined first and effort will not be wasted casting unnecessary samples within well defined regions of the image.

Finally, the ability of the new algorithm to reconstruct and filter a functional representation of the image offers many advantages over reconstruction and filtering in the spatial domain. The new algorithm is able to produce a more accurate reconstruction of the samples which can then be filtered quickly and with improved accuracy. This reduces the aliasing and noise present in the final image.

Direct synthesis into the frequency domain represents a new and powerful paradigm for computer generated imagery. This report will show how it allows knowledge from the field of image compression to be integrated into a modern rendering algorithm. This makes it possible to produce images of a superior visual quality, while reducing the total number of samples that are taken of the scene.

This report begins with a discussion of the JPEG algorithm and the advantages that this representation presents for image synthesis. This discussion is followed by a brief introduction to ray tracing and a survey of previous sampling and reconstruction schemes. The new image synthesis algorithm will then be presented. Finally, the report concludes with a discussion of the results and conclusion. The compression survey and list of image file formats that was performed in preparation for this report are contained within the appendix.

2 The JPEG Image Compression Algorithm

The JPEG still image compression standard is an excellent example of a modern image compression algorithm. It stands out for its ability to compress images and the advantages it is able to gain from the utilization of elements of human perception. The JPEG standard is the result of a joint push by ISO and CCITT to develop a standard for the representation of compressed images. The algorithm that is known as the JPEG standard won out over a large field of entries for both its ability to reduce the number of bits necessary to store an image and the perceptual quality of the image that is reproduced from those bits.

The first step in the JPEG compression of an image is to break the image into a series of 8 by 8 blocks of pixels. This is done to both increase the tractability of the later transform

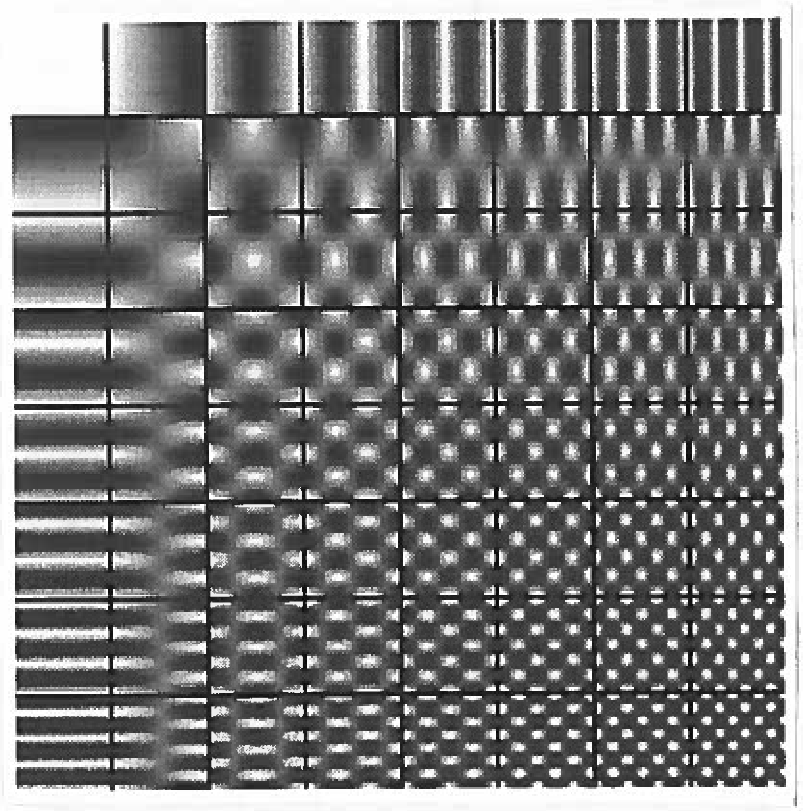

Figure 1: The basis functions of the Discrete Cosine Transform.

steps and to exploit the large correlation of intensity values found within blocks of this size. Once the image has been broken into blocks, it is then transferred into the frequency domain by the means of the Discrete Cosine Transform.

The Discrete Cosine Transform provides a means whereby a frequency domain representation of each block of the image may be obtained, in a manner similar to the Discrete Fourier Transform. This representation allows the 8 by 8 blocks of the image to be represented by an 8 by 8 sum of harmonically related cosine basis functions. These basis functions are depicted in Figure 1. The lowest frequency or "DC" term is located in the upper-left corner with the highest frequency "AC" term in the lower-right. Horizontal frequency increases from left to right and vertical frequency increases from the top to the bottom of the figure. Diagonal terms represent a combination of the horizontally and vertically affluent frequency terms.

The forward Discrete Cosine Transform is given in equation 1 below. This equation takes as input the 8 by 8 block of intensity values $f(x,y)$ from the original image and returns the 8 by 8 block of frequency coefficients $F(u,v)$. These frequency coefficients represent the magnitude of the related cosine basis functions which, when summed, will exactly reproduce the original image block. This summation is given by the reverse Discrete Cosine Transform in equation 2, which takes the frequency coefficients $F(u,v)$ and returns the original intensity values $f(x,y)$.

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

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

where

$$\begin{aligned} \kappa(u),\kappa(v) &= 1/\sqrt{2} \quad \text{for } u,v=0 \\ \kappa(u),\kappa(v) &= 1 \quad \text{otherwise} \end{aligned}$$

Figure 2 depicts a series of images built by successively adding more frequency terms to the summation of equation 2. The first image is generated by using only the DC term for each 8 by 8 block. Terms are then added to the summation in a low to high frequency manner. As the images move from left to right and top to bottom, successively more frequency terms are added to the summation. The final image is the result of using all 64 frequency terms and is an exact duplicate of the original image. Note that originally, the image is significantly blurred but basic image features are still visible. As higher frequency terms are added the image is sharpened and rapidly converges to form the original image.

The algorithm has thus far not provided any actual compaction in the number of bits necessary to represent an image. It has merely transferred the image into a different domain.

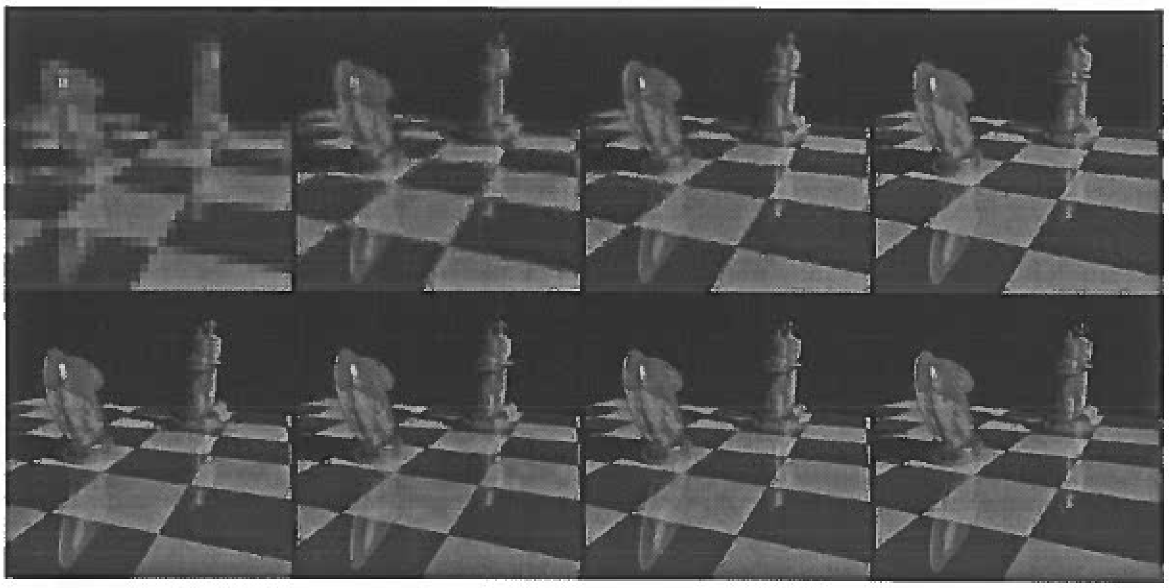

Figure 2: The images that result from using 1, 3, 6 and 10 (top row) and 15, 21, 36 and 64 (bottom row) of the lowest frequency terms within the DCT representation of the image.

A closer inspection of Figure 2, however, begins to show the means whereby this compression may result. Note that within this figure the majority of the image content is resolved with the addition of the first few frequency terms, with successive terms contributing decreasing amounts to the overall image fidelity. This effect is further illustrated in Figure 3, where an image has been decomposed into its frequency components. The 16 images contained within the figure represent the contribution of the 4 by 4 low frequency components. The contribution of the entire 64 components was not depicted due to the imperceptibility of the higher frequency terms. The images within the figure were formed by taking the magnitude of individual frequency components from each block and combining them to form an image $(\frac{1}{64})^{th}$ the size. The ordering of the images is related to the frequency term being represented and is the same as the ordering of the basis functions given in Figure 1. The magnitudes of the coefficients can be seen to decrease dramatically as the frequency increases and become virtually imperceptible for the higher frequency components. This ability to represent the majority of the energy of the image within the first few low frequency terms is, in fact,

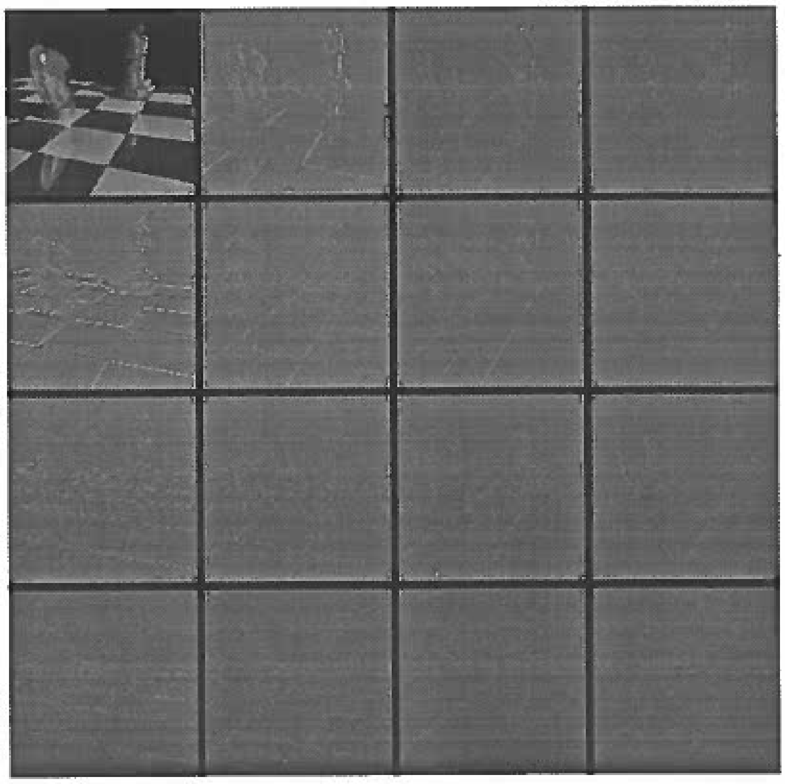
Figure 3: Magnitude of the low 4 by 4 DCT frequency components within the chess board image.

an inherent property of the Discrete Cosine Transform. Rao and Yip (1990) have further shown that the DCT is asymptotically equivalent to the optimal Karhunen-Loeve transform in terms of this energy packing efficiency. This low frequency energy prevalence implies that the majority of the higher frequency terms will, in general, be small or non-existent. This imparts a large amount of redundancy to the space which is exploited by the later steps of the compression algorithm.

The JPEG compression algorithm is inherently a lossy compression method (although a loss-less mode is available as well). It had as one of its design goals, to provide for an adjustable quality to compression trade-off. This trade-off is obtained by varying the accuracy with which the frequency coefficients are represented, through the use of a quantization step. This quantization could be performed uniformly across all frequency coefficients and some measure of compaction would result, due to the redundancy of the frequency space. The algorithm, however, is able to greatly improve the perceived image quality at a given compression level by noting that human visual acuity has a frequency dependent nature. The human visual system is able to perceive changes in intensity much more readily at low frequency oscillations than at high frequencies. This allows the algorithm to represent the higher frequency terms much less accurately than the low frequency terms without visibly impairing the perceived image quality.

The visual importance of the high frequency terms is further reduced by an effect known as "masking". Masking is an effect whereby the visibility of noise is greatly reduced in areas of high spatial detail. This is a result of the fact that the sensitivity of the retinal receptors can be greatly affected by the presence of other visual stimuli within the immediate spatiotemporal neighborhood (Schreiber 1993). In this case, the presence of high frequency stimuli can hide or "mask" the presence of quantization noise within the reconstructed image. This effect is illustrated in Figure 4, where low frequency (upper-left) and high frequency (upper-right) fractal mountainsides have been modeled. In the bottom row the intensity values of the original images have been quantized to 4 bits per channel. The banding caused by this quantization is clearly visible on the low frequency mountain (lower-left). However, the effect of this quantization is much more difficult to distinguish on the high frequency mountain (lower-right), which is virtually indistinguishable from the original high frequency image.

In order to take advantage of the greater importance of the low frequency coefficients, both in terms of energy and visual significance, a quantization table is employed by the JPEG algorithm. A sample quatization table is depicted in figure 5.

This table describes the quantizer step size $Q(u, v)$, which will be applied to the fre-

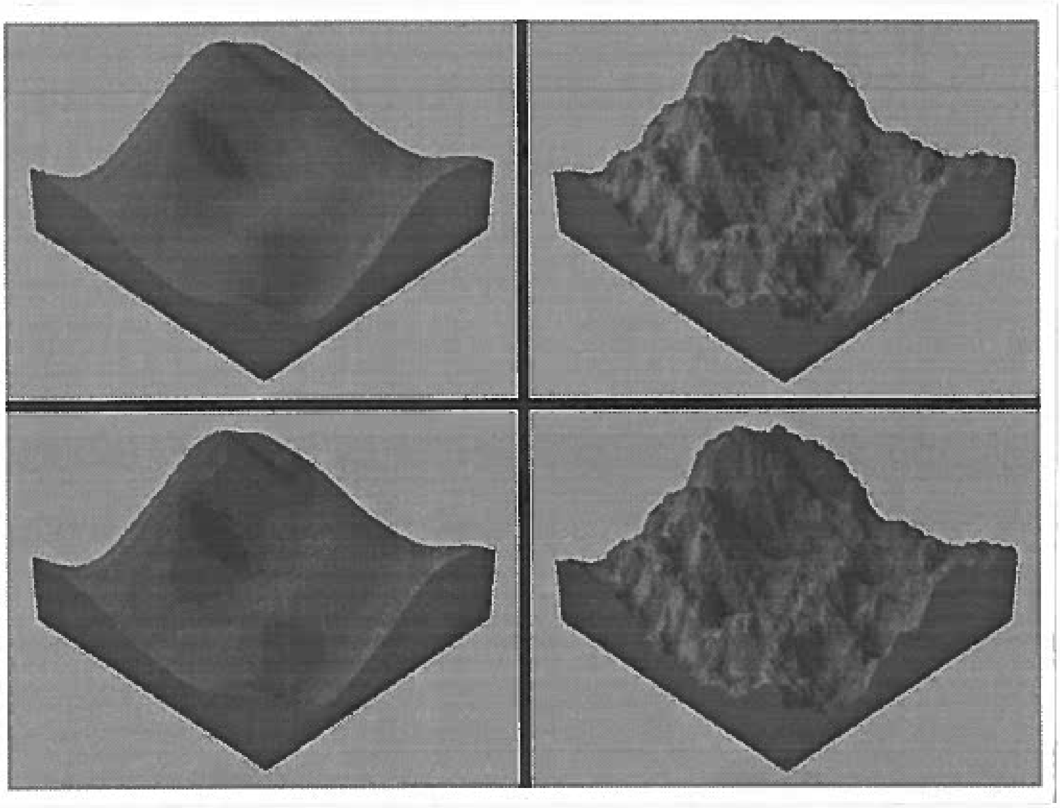
Figure 4: Visibility of noise when images with different spatial frequency content (top row) are quantized to 4 bits (bottom row).

quency coefficients to determine the final coefficients $F^Q(u,v)$ to be encoded. The formula that is employed to quantize the coefficients is given in equation 3.

$$F^Q(u,v) \quad = \quad \text{Integer Round}\left[\frac{F(u,v)}{Q(u,v)}\right] \tag{3}$$

The actual quantization table used varies depending upon the desired image quality. This quantization step allows the low frequency coefficients to be represented with greater accuracy than the higher frequency coefficients. The majority of the high frequency coefficients will, in fact, tend to be zero due to the minimal energy contained within these terms and the larger quantization step size applied to them. In this manner, the redundancy of the frequency coefficients is increased, especially in the higher frequency terms.

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 5: A sample quantization table (from Wallace (1991)).

Once the frequency coefficients have been quantized they are then ordered according to the "zig-zag" sequence given in figure 6. This ordering helps to facilitate the following entropy encoding step by placing low frequency coefficients (which are more likely to be non-zero) before high frequency coefficients. This ordering additionally allows higher frequency terms which are more horizontally or vertically affluent to precede some of the lower frequency diagonal terms. The horizontal and vertical selectivity of the topmost row and leftmost column of frequency terms may be seen upon inspection of the associated row and column within Figure 3. This ordering is performed to take into account the greater prevalence of horizontal and vertical elements within most common images. In addition, these horizontal and vertical frequency terms are represented with a smaller quantization step size to take into account the greater sensitivity of the human visual system to horizontally and vertically aligned intensity contours. These elements will thus tend to have a larger, less redundant magnitude and should be placed earlier in the ordering sequence.

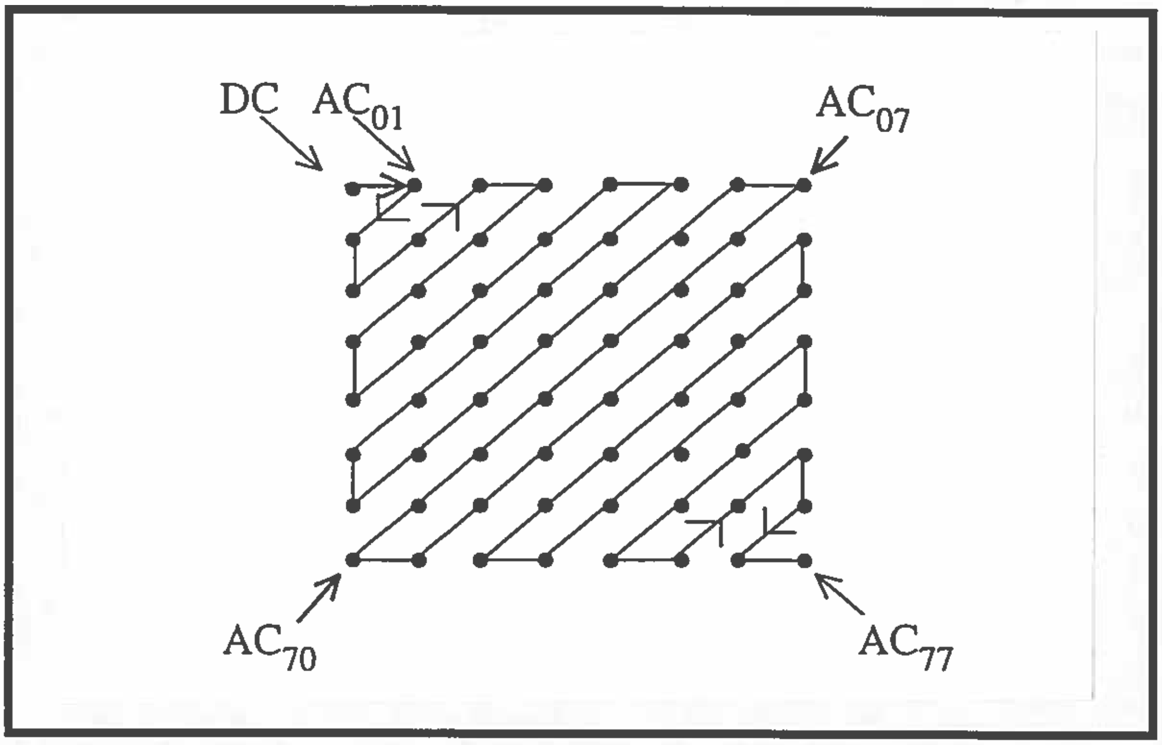

Figure 6: The zig-zag sequence (from Wallace (1991)).

Once the coefficients have been placed in this order, the redundancy in the frequency representation that occurs as a result of the low frequency energy prevalence and quantization must be removed. This is accomplished through a combination of Differential Pulse Code Modulation, entropy encoding (either Huffman or Arithmetic) and run-length encoding techniques. The output of these techniques forms the final bit stream that is stored as the compressed representation of the image. These final techniques will not be described in detail here but may be found in the appendix or in an article by Wallace (1991).

The JPEG algorithm can thus be seen to take unique advantage of the frequency domain representation given by the Discrete Cosine Transform. This representation allows the algorithm to benefit from both the energy distribution of the frequency terms and the perceptual significance of those terms. This allows the algorithm to create a more redundant space in which an image may be significantly compressed while retaining a high level of perceptual quality in the reconstructed image.

3 Ray Tracing Theory

The theory of ray tracing was first proposed by Appel (1968) who used his algorithm to compute shadows and shading and by Goldstein and Nagel (1971 and MAGI 1968) who used their algorithm to simulate the trajectories of ballistic projectiles and nuclear particles. Modern ray tracing as an image synthesis technique is generally deemed to have begun with Kay (1979) and Whitted (1980) who suggested the first general ray tracing paradigm and extended ray tracing to include specular reflection and refraction.

Ray tracing is an image synthesis technique whereby the interaction of light with a scene is computed for a specific viewpoint. Rays are cast "backwards" from the eye point through the pixels of the view plane and intersected with the model that composes the scene. Once the ray intersects a surface, a shader is evaluated to determine the color of light at that point on the surface.

The most commonly used local shading model is known as the Phong model (Bui-Tuong 1975). This model computes the intensity of light at a surface as the combination of ambient, diffuse and specular terms. The ambient term represents a "fudge factor" necessary to account for the surface illumination that results from extraneous non-directional light present in the scene. It is given by the product of a constant ambient intensity for the scene and the ambient reflectance coefficient of the surface. The diffuse term represents the matte reflectance of the surface, meaning the amount of light that is reflected equally in all directions. This term is computed from the angle between the light source(s) and the normal to the surface times the diffuse coefficient of the surface. The specular term indicates the illumination that occurs as a result of the shiny reflectance of the surface. It is computed from the angle between the view vector and the vector given by the reflection direction of the light from the surface. This specular term is multiplied by a specular coefficient of the surface and a raised cosine term representing the strength of the specular highlight.

Whitted (1980) extended this model to include reflection and refraction and presented the idea of using a ray tree to compute global illumination terms. Using his approach additional rays are spawned in the reflected and transmitted directions to compute the illumination from these sources. These rays are then intersected with the scene and the algorithm recurses until a fixed depth or attenuation factor has been reached.

The weakness of this approach for image synthesis is that it is not able to represent the more complex reflectivities of some surfaces and it is not able to simulate the more involved interactions of light such as diffuse-diffuse, specular-diffuse and so forth. A technique used to model these types of interaction is known as Monte Carlo ray tracing (Kajiya 1986). In this technique a bi-directional reflectance distribution function (BRDF) is associated with a surface. This BRDF specifies the reflectivity of the surface as a function of incident angle, reflected angle and wavelength. When a ray is intersected with this surface, rays are spawned which sample all directions of the reflectance function and the incident light from the scene at that direction. These rays then recurse through the scene until the stopping criteria is met. This process is inherently expensive to compute and generally noisy in its result. Care must be taken to sample the scene as efficiently as possible and a good reconstruction filter must be applied in order to obtain smooth results. The technique used to model the bi-directional reflectance distribution function within this report is based on a technique proposed by Gondek, Meyer and Newman (1984) which utilizes a geodesic sphere and novel data structure.

3.1 Survey of Sampling and Reconstruction Techniques

3.1.1 Sampling the Scene

Ray tracing is inherently a point sampling problem. The two-dimensional image plane is sampled at discrete locations to determine intensity values which are then filtered and resampled at the pixel locations to reconstruct the desired image. All point sampling problems are bound by the Nyquist criteria which says that a signal must be sampled at twice the highest frequency present within that signal in order to reproduce the signal without aliasing. Aliasing implies that high frequency components of the signal are masquerading as low frequency components within the reconstructed signal. In ray tracing undersampling produces aliasing in the form of "jaggies" or jagged intensity contours where the sample from one pixel may have fallen to one side of the contour and the sample from an adjoining pixel may have fallen to the other. Although the contour may gradually transition through these pixels the reconstructed image shows a harsh discontinuity between the neighboring pixel's intensities.

One technique used to overcome aliasing in the rendering of images is super-sampling or, the casting of many rays into each pixel in an effort to meet the Nyquist criteria for the highest frequency portion of the image. The disadvantage of this technique is that it is rather expensive and samples both high and low frequency regions of the image at a high rate resulting in wasted effort and increased computation time.

A better method is to allow the sampling rate to adjust to fit the frequency content of the underlying image. This technique is known as adaptive sampling. This technique requires a metric that is capable of giving an estimate of the underlying frequency content of the scene. The majority of previous works in this area have used a statistical measure of the variance in the intensity returned by the image samples to provide this estimate.

An alternate method was presented by Dippe' and Wold (1985), who suggested that the difference in the reconstructed images at varying sample rates be used rather than the samples themselves, to provide this estimate.

Three basic methods have been developed to provide adaptive sampling of a scene. The first was presented by Whitted (1980), who described a pixel based method. In this method the pixels of the image are traversed in order. At each pixel a sample is taken through each of the four corners. If these samples exhibit a high variance or if a small object exists between them, the pixel is sub-divided and the process repeats. The disadvantage of this method is that it is a local method, wherein the entire image must be sampled completely before an image may be constructed. However, this method has the advantage that only the number of samples taken within a single pixel need to be stored at any given time.

The second method begins to provide the means whereby the image may be presented to the user as it is being refined. Variants of this method have been utilized by both Cook (1986) and Mitchell (1987). It provides for a limited form of adaptation with some discrete number of levels of sampling densities (typically 2). In this method the image plane is divided into a number of fixed size cells. These cells are then sampled at an initial sampling density and the variance of the samples is computed. The cells which exhibit high variance are then resampled at a greater density. This process may then optionally recurse. If an iterative solution is not required, a fixed variance threshold may alternatively be set, which all cells must fall below. This allows the cells to be processed in order and eliminates the need to store samples across the entire scene. However, this technique does not allow for refinable user feedback. The main advantage of dividing the image plane into fixed size cells is that it allows greater flexibility in the choice of sampling patterns. This results since the sampling pattern is of fixed size and density and therefore may be readily pre-computed.

In addition this fixed size and density allows for easier computation of variance thresholds and reconstruction filters. This method has the disadvantage, however, that it only allows for fixed levels of sampling densities, with only limited sampling adaptation between cells.

The third technique is the only one that provides true adaptive sampling across the entire image plane. It was proposed by Painter and Sloan (1989). This technique creates a tree which contains leaves consisting of all samples taken of the image. At each node of the tree the variance is calculated of all samples which lie below the node. To determine where the next sample should be taken, the tree is traversed beginning at the root. At each node a branch is chosen based on the variance, the area of the node, the number of samples already contained in the node and the level of the node in the tree. At the upper levels of the tree the decision is weighted toward providing the best coverage (i.e. large nodes are chosen more often than small ones). In the lower levels of the tree the decision is instead weighted toward edge detection, with the magnitude of the variance dominating the decision. When a leaf node is reached, it is split, a new sample cast and the variances re-calculated up the tree. This method provides excellent results and a continuum of adaptation across the image. The main downfall of this technique is the expense involved in creating and maintaining the sample tree.

Numerous researchers have proposed that a more visually pleasing image may be produced through the use of a Poisson or jittered sampling distribution, often in conjunction with the above techniques. A Poisson distribution is formed by a random distribution of samples such that no two samples are closer than a fixed threshold. This distribution is expensive to compute and is more often approximated by a jittered distribution which is formed by randomly displacing samples a limited distance from a regular grid. These distributions have the advantage that they can trade the sharp aliasing which occurs in regular sampling patterns for high frequency noise. This noise is generally deemed less objectionable to the human visual system.

3.1.2 Reconstructing the Image

Once the scene has been sampled, an image must be reconstructed from these samples. Ideally this would result by reconstructing a representation of the underlying image from the samples, filtering this representation to remove aliasing and bandlimit the signal to the display resolution and finally, resampling this representation to obtain the pixel intensities. This process, however, involves a number of difficulties and currently has only been crudely approximated within the existing algorithms.

The first such difficulty is a result of the fact that previously all samples must be stored in order to reconstruct and filter them later. This is especially expensive for non-local techniques such as tree-based adaptive subdivision, which must store the potentially millions of samples that occur across the entire image plane.

In addition, filtering the reconstructed image is an expensive process within the spatial domain. This results from the fact that filtering in the spatial domain involves the convolution of a filter function with the reconstructed image. This convolution is an expensive process, especially if the image is being iteratively refined, since the image must be repetitively reconstructed and filtered as the image resolves.

Finally, the process of reconstructing and filtering a series of irregularly spaced samples is difficult and not well understood. Irregularly spaced samples result from using Poisson or jittered sample distributions or from any iterative sampling scheme. Reconstruction of a function from a series of irregularly spaced samples is a difficult process that has previously been given by only crude approximations. Alternatively, discrete methods have been employed to filter individual samples directly. This method only gives a rough approximation of the filter convolution step. However, the difficulty involved in reconstructing the samples and the increased speed and tractability of this method, have made this method an attractive choice. The difficulty involved with filtering irregularly spaced samples directly is that a variable number of samples may occur underneath the filter at variable spacings, which can result in what is termed "grain noise". Grain noise is an artifact which results from the filter being abnormally weighted toward areas of high sample density.

A number of methods have been employed to reconstruct an image from its samples. The first and crudest method determines the intensity of a pixel by the simple average of the samples that occur within it. If the samples are irregularly spaced, they are often weighted by the area that the sample is deemed to define. This process is known as box-filtering and produces numerous undesirable artifacts.

Alternatively Painter and Sloan (1989) proposed that the image be divided into cells, each of which contain exactly one sample. The intensity is defined to be constant across the cell and given by the sample that lies within it. This representation of the image may then be convolved with more advanced filters by integrating the filter over the area of the cell and multiplying it by the intensity of the sample. For simple filters this integral may be computed analytically but, for more advance filters, they propose the use of a "summed area table" in order to determine the value of the integral. This method is nice since it allows the use of superior filters. However, the piecewise constant representation of the image may be deemed crude at best.

The use of weighted average filters has been proposed by several authors as a means to filter discrete samples directly. In this method the intensity of a pixel is defined to be the sum of the samples within that pixel's filter radius, times the value of the filter at those sample positions. This value is then normalized by dividing by the sum of the values of the filter at all sample positions within the given radius. This method is nice as it allows advanced filters to be applied directly to sample values. It suffers however, as a result of the "grain noise" discussed earlier, which can be produced by the clumping of samples within sub-regions of the filter area.

Mitchell (1987) proposed a solution to this problem through the used of multi-stage filters. In this method weighted average filters are applied to successively larger filter radii, with each operating on the output of the prior filter. In this manner dense clumps of samples will be weighted equally with more sparse clumps of samples, and grain noise will be reduced. The only major downside to this technique is the additional cost accrued by the multiple convolutions of the filter function that is now necessary.

Dippe' and Wold (1985) further enhanced the use of weighted average filters by suggesting the image could be adaptively filtered by allowing the filter width to vary based on the local signal to noise ratio. Thus allowing regions with a high SNR to be significantly filtered, with less filtering occurring in regions of low SNR. This technique is advantageous since it allows the noise prevalent in certain regions of the image to be reduced without the use of excessive filtering across the image, which could significantly blur other well defined and salient features of the image.

3.1.3 Post-Filtering the Image

An additional technique has been proposed to further reduce noise in the output image. This technique involves post-filtering the image with non-linear filters to smooth noisy regions of the image. Large amounts of noise can be created in an image through the use of Poisson or jittered sampling distributions, Monte Carlo ray tracing and, are always the result of inadequate reconstruction and filtering of the original samples.

Lee and Redner (1990) proposed the use of alpha-trimmed non-linear filters as a means to smooth the final image. In this technique a series of hexagonally spaced samples are generated around each pixel position by bi-linearly interpolating from the neighboring four pixel intensities. A user specified fraction of the largest and smallest intensities are then discarded and the average of the remaining samples are computed to yield the final pixel intensity. In this way the noisiest samples are discarded and the resultant image can be greatly smoothed.

Rushmeir and Ward (1994) proposed an alternate technique that is energy preserving. They state that since all pixels contain valid information, this information should not be discarded. Instead, they propose a technique whereby noisy pixels are identified and the excess noise is distributed to surrounding pixels. The width and magnitude of this distribution is determined by the variance of the pixel from the average of its surrounding neighbors, and is distributed such that no pixel receives more than a fixed threshold of the excess energy. In this manner no information is discarded, it is instead merely redistributed to its surrounding neighbors. This effectively low-pass filters the image noise in an adaptive manner similar to that of Dippe' and Wold, with the additional benefit that absolute radiance is preserved across the entire image.

The frequency domain synthesis technique presented in the following section will be shown to offer similar noise reduction properties to the algorithm presented above. In addition it will provide for a novel means whereby the scene may be adaptively sampled, reconstructed and filtered. This will further be accomplished without the need for the expensive convolution and post-processing steps required by the existing algorithms.

4 Synthesis Into the Frequency Domain

The frequency domain representation into which we shall synthesize images forms the first stage of the JPEG compression algorithm. At this stage the image has been broken into a series of 8 by 8 pixel blocks and the Discrete Cosine Transform has been applied in order to convert these intensity values into the frequency domain. This frequency domain representation is given by a series of 8 by 8 blocks of frequency coefficients. These coefficients represent the magnitudes of the cosine harmonics necessary to reconstruct the image within a specific block. Separate sets of frequency coefficients are necessary to reconstruct each of the color channels present within an image. An iterative and adaptive least squares technique is presented that is capable of generating and progressively refining this frequency domain description of the image. This technique will be shown to offer a novel solution to the sampling and reconstruction problem involved in ray tracing a synthetic scene.

4.1 Computing the Frequency Representation

The iterative conversion of a block of samples into the frequency domain provides the basis of a method whereby the representation of an image may be gradually refined from its samples. However, this iterative conversion of samples cannot be achieved by a simple application of the Discrete Cosine Transform. This problem stems from the fact that the DCT requires as input 2^K uniformly spaced samples, which is not possible to provide in any iterative sampling scheme. Fortunately, the format of the inverse Discrete Cosine Transform (Equation 2 in Section 2 above) allows the solution of the frequency domain coefficients to be viewed as a least squares problem. In this manner, these coefficients will form the unknown variables which will gradually be refined in an iterative curve fitting manner.

4.1.1 Least Squares Curve Fitting

The least squares process provides us with a method whereby the unknown variables of certain classes of functions may be determined such that the error between the sampled values and the reconstructed curves are at a minimum. This is the method we shall use in order to obtain the frequency coefficients which define the DCT Transform. This development is based on the discussion of least squares curve fitting given by Stark (1970).

From the inverse DCT of Equation 2 we see that

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

Let the subscript i impose an arbitrary ordering on the frequency terms u and v given by u_i and v_i. This equation then becomes

$$f(x,y) = \frac{1}{4}\sum_{i=0}^{63}\kappa(u_i)\kappa(v_i)F(u_i,v_i)\cos\left[\frac{(2x+1)u_i\pi}{16}\right]\cos\left[\frac{(2y+1)v_i\pi}{16}\right]. \qquad (5)$$

To simplify the notation allow

$$G(u_i,v_i) = \frac{1}{4}\kappa(u_i)\kappa(v_i)F(u_i,v_i) \qquad (6)$$

and let

$$C_{\alpha,\beta} = \cos\left[\frac{(2\alpha+1)\beta\pi}{16}\right]. \qquad (7)$$

This results in the equation

$$f(x,y) = \sum_{i=0}^{63} G(u_i,v_i)C_{x,u_i}C_{y,v_i}. \qquad (8)$$

Let $S(x_n, y_n)$ be a sample n, taken at the point (x_n, y_n). Then the error in the approximation of sample n is given by

$$\begin{aligned} \delta_n &= G(u_0, v_0)C_{x_n,u_0}C_{y_n,v_0} + G(u_1, v_1)C_{x_n,u_1}C_{y_n,v_1} + \cdots + \\ &\quad G(u_{63}, v_{63})C_{x_n,u_{63}}C_{y_n,v_{63}} - S(x_n, y_n) \qquad (9) \\ &= f(x_n, y_n) - S(x_n, y_n). \qquad (10) \end{aligned}$$

The least squares solution for N samples is given when the following equation is at a minimum:

$$\sum_{n=0}^{N} (\delta_n)^2. \qquad (11)$$

This summation may also be written as

$$\sum_{n=0}^{N} [G(u_0, v_0)C_{x_n,u_0}C_{y_n,v_0} + G(u_1, v_1)C_{x_n,u_1}C_{y_n,v_1} + \cdots + \\ G(u_{63}, v_{63})C_{x_n,u_{63}}C_{y_n,v_{63}} - S(x_n, y_n)]^2 . \qquad (12)$$

Note that the terms C_{x_n,u_i}, C_{y_n,v_i} and S_{x_n,y_n} are all known values with only the 64 coefficients $G(u_i, v_i)$ being unknown. To minimize a function we set its first derivative to 0 and solve for the unknown variable. In this case we have a function of 64 unknown variables. We therefore take the partial derivative with respect to each of the coefficients $G(u_j, v_j)$ and set each of these equal to 0:

$$\frac{\partial}{\partial G(u_j, v_j)} \sum_{n=0}^{N} (\delta_n)^2 = 0 \qquad \text{for } j = 0, 1, \ldots, 63. \qquad (13)$$

Using the fact that a derivative of a sum is equal to the sum of the derivatives, we have

$$\frac{\partial}{\partial G(u_j, v_j)} \sum_{n=0}^{N} (\delta_n)^2 = \sum_{n=0}^{N} \frac{\partial}{\partial G(u_j, v_j)} (\delta_n)^2 = \sum_{n=0}^{N} 2\delta_n \frac{\partial \delta_n}{\partial G(u_j, v_j)} = 0 \tag{14}$$

or

$$\sum_{n=0}^{N} \delta_n \frac{\partial \delta_n}{\partial G(u_j, v_j)} = 0. \tag{15}$$

Given the linear independence of the basis functions, we note that all terms in the summation of equation 9 are constant with respect to a specific coefficient $G(u_j, v_j)$ except the term containing that coefficient. Differentiation thus becomes a simple process since all other terms drop out upon differentiation. This derivative is given by

$$\frac{\partial \delta_n}{\partial G(u_j, v_j)} = 0 + 0 + \cdots + \frac{\partial}{\partial G(u_j, v_j)} G(u_j, v_j) C_{x_n, u_j} C_{y_n, v_j} + 0 + \cdots + 0, \tag{16}$$

which reduces to

$$\frac{\partial \delta_n}{\partial G(u_j, v_j)} = C_{x_n, u_j} C_{y_n, v_j}. \tag{17}$$

Substituting into equation 15 we have the 64 equations

$$\sum_{n=0}^{N} \delta_n C_{x_n, u_j} C_{y_n, v_j} = 0 \qquad \text{for } j = 0, 1, \ldots, 63. \tag{18}$$

Substituting from equation 9 and multiplying through we have

$$\sum_{n=0}^{N}\Big[G(u_0,v_0)C_{x_n,u_0}C_{y_n,v_0}C_{x_n,u_j}C_{y_n,v_j}+\cdots+G(u_{63},v_{63})C_{x_n,u_{63}}C_{y_n,v_{63}}C_{x_n,u_j}C_{y_n,v_j}-$$
$$S(x_n,y_n)C_{x_n,u_j}C_{y_n,v_j}\Big]=0 \qquad \text{for } j=0,1,\ldots,63. \quad (19)$$

Distributing the summation and reorganizing yields

$$G(u_0,v_0)\sum_{n=0}^{N}C_{x_n,u_0}C_{y_n,v_0}C_{x_n,u_j}C_{y_n,v_j}+\cdots+G(u_{63},v_{63})\sum_{n=0}^{N}C_{x_n,u_{63}}C_{y_n,v_{63}}C_{x_n,u_j}C_{y_n,v_j}$$
$$=\sum_{n=0}^{N}S(x_n,y_n)C_{x_n,u_j}C_{y_n,v_j} \qquad \text{for } j=0,1,\ldots,63. \quad (20)$$

Thus we finally have the desired form, consisting of 64 equations and 64 unknowns. This may be rewritten in matrix format as

$$[J]\,[X] \;=\; [Y]\,. \tag{21}$$

Where X and Y are given by 64×1 column vectors and, J forms a 64×64 matrix. The terms of which are given by

$$J_{ij} = \sum_{n=0}^{N}C_{x_n,u_i}C_{y_n,v_i}C_{x_n,u_j}C_{y_n,v_j} \tag{22}$$
$$Y_i = \sum_{n=0}^{N}S(x_n,y_n)C_{x_n,u_i}C_{y_n,v_i} \tag{23}$$
$$X_i = G(u_i,v_i). \tag{24}$$

The solution of this least squares problem is then given by inverting the J matrix and multiplying through on both sides. This yields the equation

$$[X] = [J]^{-1}[Y]. \tag{25}$$

The frequency coefficients can then be obtained by solving for X which yields the terms $G(u,v)$ which can be converted to the original frequency coefficients by inverting equation 6:

$$F(u,v) = \frac{4}{\kappa(u)\kappa(v)}G(u,v). \tag{26}$$

The discussion of this and ongoing sections is based on the synthesis of monochromatic samples into a frequency domain representation. In actuality, the sampling of a synthetic scene generally returns three or more samples $S(x_n, y_n)$, one for each color channel which forms the representation of the image. In this case separate X and Y vectors must be formed and a separate frequency representation refined for each color channel. Note should be taken, however, that the matrices J and J^{-1} are independent of the number and content of the color channels, and need be computed only once. The single matrix J^{-1} may then be multiplied by the Y matrices formed for each of the color channels in order to compute the frequency representation for each of the channels. This is advantageous since the computation of the inverse matrix J^{-1} will form the bulk of the computation necessary for the algorithm presented within this report.

4.1.2 Iterative Matrix Solution

One of the goals of synthesizing images into the frequency domain was to provide for an iterative means whereby image content may be gradually resolved as more samples are taken of the scene. A closer inspection of the results of the previous section provides us with a means whereby this result may be obtained.

Equation 25 tells us that the estimate of the frequency coefficients $F(u,v)$ that best fits the existing N samples may be obtained by multiplying the inverse matrix of J by the column vector Y. However, if we now take our $(N+1)^{th}$ sample a new term must be added to the summation of equation 22 and equation 23. The new matrices J' and Y' are then formed by the addition of a single term to each existent entry in both matrices. The new estimate of the frequency coefficients may then be obtained by inverting the new matrix J' and multiplying it by the new matrix Y'.

This is a somewhat costly operation, however, since the inverse of a matrix takes $O(N^3)$ time to compute. Fortunately, we can do better. The new entries of matrix J' are given by

$$J'_{ij} = J_{ij} + C_{x_{N+1},u_i} C_{y_{N+1},v_i} C_{x_{N+1},u_j} C_{y_{N+1},v_j}. \tag{27}$$

If we define a 64×1 column vector A to have entries given by

$$A_i = C_{x_{N+1},u_i} C_{y_{N+1},v_i}, \tag{28}$$

we may recast equation 27 into matrix form. This yields the result that

$$J' = J + AA^T \tag{29}$$

and

$$(J')^{-1} = (J + AA^T)^{-1}. \tag{30}$$

This forms what is termed a rank-one inverse matrix modification problem. Powell (1969) presented a solution to problems of this type that is tuned to suppress the accumulation of errors. He proved that inverse matrices of the type

$$(J')^{-1} = (J + uv^T)^{-1}, \tag{31}$$

may be solved by the equation

$$(J + uv^T)^{-1} = J^{-1} + \frac{(v - J^{-1}\gamma)v^T J^{-1}}{(v^T J^{-1}\gamma)} \tag{32}$$

where

$$\gamma = u(v^T v) + Jv.$$

There is no restriction that the entries of u and v^T must differ. Thus, when we apply this theorem to our problem we get the result that

$$(J')^{-1} = J^{-1} + \frac{(A - J^{-1}\gamma)A^T J^{-1}}{(A^T J^{-1}\gamma)} \tag{33}$$

where

$$\gamma = A(A^T A) + JA.$$

The solution of which may be computed in $O(N^2)$ time. Thus we have produced a method where we may iteratively sample a scene and apply a least squares solution in order to progressively refine the frequency representation of the image. This is done with a method which may be computed in a reasonable amount of time and with little induced error.

4.2 Adding Additional Frequency Terms

The formula given in equation 33 of the proceeding section describes a method whereby samples of a synthetic scene may be iteratively converted into a frequency domain representation. However, this formula is only stable if the inverse matrix J^{-1} is actually computable.

This implies that in order for equation 33 to be accurately applied, the matrix J must be non-singular. This condition is only satisfied when all of the rows and columns of J are linearly independent. In terms of representing the image with a series of harmonic cosine waves, this means that the coefficients of the cosine terms must not be able to be specified in more than one manner in order to yield the minimum possible error. This only occurs when enough samples have been taken to uniquely represent all of the frequency terms present in this equation. This problem is easy to visualize at the point a single sample has been taken of the scene, since this sample can be accurately fit with any one of the frequency terms with zero error.

Since we desire an iterative solution to the frequency domain synthesis problem, it is undesirable to wait until enough samples have been taken to uniquely represent all 64 frequency terms before we can begin the iterative process. It would be preferable if we could begin iterative synthesis using only enough frequency terms that are justified given the current amount of sampling, and increase the number of frequency terms to be solved for as the number of samples justifies it. Reference to the development of the previous section shows that although the formulas were presented for the use of 64 frequency terms, no special use was made of this fact. An 8 by 8 block of intensity values may be represented with less frequency terms by the following modification of equation 5:

$$f(x,y) \quad = \quad \frac{1}{4}\sum_{i=0}^{T-1} \kappa(u_i)\kappa(v_i)F(u_i,v_i)C_{x,u_i}C_{y,v_i} \tag{34}$$

where

$$T \leq 64.$$

The development of the last section then proceeds exactly as before, although T frequency terms will be used instead of 64. This yields the same results in equation 25 and equation 33 except that X, Y and A are now $T \times 1$ vectors, and J forms a $T \times T$ matrix. These factors are all just submatricies of the original 64 term matrices. The frequency coefficients that form the vector X may then be iteratively solved for as before by inverting the matrix J and multiplying it by the column vector Y. One should note that initially this operation may be performed quite rapidly, since the early stages of iteration involves only a small number of frequency terms.

As the quantity of samples taken within a given block increases, it becomes possible to add additional frequency terms to the calculation. This implies that the vectors X, Y and A must be expanded as well as the matrices J and J^{-1}. The vectors X, Y and A may all be expanded by simply adding an additional row, and the matrix J may be expanded by adding an additional row and column. The entire 64 term versions of X, Y, A and J are stored to allow this operation to be performed without the need for storing all sampled values. The matrix J^{-1}, however, cannot be expanded by the simple addition of a pre-existing row and column, since the singularity of the matrix J prevents the 64 term inverse matrix from being pre-calculated. In addition we cannot use equation 33 to calculate this expanded inverse since it would require a $(T+1)$ sized J^{-1} from which to begin iteration. Finally, it is undesirable to invert the expanded matrix J by conventional means since this involves an $O(N^3)$ operation, which would have to be performed up to 64 times during the progression of the algorithm.

Fortunately, a more efficient method is available. Frobenius' relation (Bodewig 1956) describes a method whereby the inverse of an expanded matrix may be computed in $O(N^2)$ time given the inverse of a smaller sub-region of that matrix. It is

$$P^{-1} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} R & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} RB\Delta^{-1}CR & -RB\Delta^{-1} \\ -\Delta^{-1}CR & \Delta^{-1} \end{bmatrix} \tag{35}$$

where

$$\begin{aligned} R &= A^{-1} \\ \Delta &= D - CRB. \end{aligned}$$

If we let the subscript T and $T+1$ denote the vector or matrix consisting of T or $T+1$ terms respectively, then we may represent the expanded matrix J_{T+1} as

$$J_{T+1} = \begin{bmatrix} J_T & B \\ B^T & D \end{bmatrix}. \tag{36}$$

Where B forms the $T \times 1$ column vector with terms given by

$$B_i = \sum_{n=0}^{N} C_{x_n,u_i} C_{y_n,v_i} C_{x_n,u_{T+1}} C_{y_n,v_{T+1}}.$$

And D forms the single entry given by

$$D = \sum_{n=0}^{N} \{C_{x_n,u_{T+1}} C_{y_n,v_{T+1}}\}^2.$$

Applying Frobenius' relation to our problem implies that we may add an additional frequency term and compute the expanded matrix $(J_{T+1})^{-1}$ through the relation

$$(J_{T+1})^{-1} = \begin{bmatrix} (J_T)^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} (J_T)^{-1}B\Delta^{-1}B^T(J_T)^{-1} & -(J_T)^{-1}B\Delta^{-1} \\ -\Delta^{-1}B^T(J_T)^{-1} & \Delta^{-1} \end{bmatrix} \tag{37}$$

where

$$\Delta = D - B^T(J_T)^{-1}B.$$

Use was made of the redundancy of some of the above calculations and an ordering was imposed on the calculation of terms to further reduce the number of multiplications necessary to perform this calculation.

Thus, the techniques presented in the previous section can be employed to iteratively refine the frequency representation of an image, starting originally with a single frequency term. At the point that it is deemed that a new term should be added, the matrix J should first be inverted using the original number of terms. The inverse matrix may then be expanded through the use of equation 37. The vectors X, Y and A and matrix J may be expanded by a simple addition of terms. Finally, equation 25 may be used to solve for the expanded frequency coefficients. The algorithm may then return to the techniques of the previous section and continue to iterate in order to refine the new and expanded frequency representation of the image. This process repeats as necessary, until all 64 frequency terms are added. At which point no new terms will be added but, the image representation may continue to be refined using the techniques of the previous section.

4.2.1 Ordering the Frequency Terms

Until now the ordering of the frequency terms within the vectors X, Y and A and the matrix J has been left arbitrary. Thus, the order in which these terms are added to the representation of the image and iteratively refined has also been left arbitrary. However, certain sequences in the addition of frequency terms make more sense than others.

For instance, when only a single sample has been taken within a given 8 by 8 block it does not make sense to fit a high frequency term to this sample, as this would result in a wildly oscillating intensity variation across this region of the image, which has little justification. Instead it makes the most sense to fit this sample with the lowest frequency or "DC" term, giving a constant intensity across this region of the image at the intensity value returned by the sample. This observation holds in general, in that, we wish to add frequency terms to our equation in a low to high frequency manner to produce the most visually pleasing images as the algorithm progresses. This observation is borne out by the Nyquist theorem which states that a signal must be sampled at twice the highest frequency present in that signal in order to accurately reconstruct it. Thus, higher sampling rates are necessary to accurately reproduce higher image frequency content.

The JPEG algorithm provides such a low to high frequency ordering for the two dimensional frequency terms present within our representation. This is given by the zig-zag ordering present within the JPEG algorithm. Although this order does not strictly place all low frequency terms before all higher frequency terms, it does have some preferable qualities. This ordering of terms allows some of the more horizontally and vertically affluent frequency terms to precede some of the lower frequency diagonal terms. This is preferable both within the JPEG algorithm and within this image synthesis technique, in that it allows the algorithms to concentrate on, and more accurately represent the horizontal and vertical frequency terms. This correlates well with the anisotropic spatial frequency response of the human visual system. In addition, since we are trying to synthesize images into the first stage of the JPEG compression algorithm, it behooves us to concentrate first on the frequency terms to which the JPEG algorithm will assign the most bits.

Therefore, we shall let the ordering placed on the vectors X, Y and A as well as the matrix J be that given by the zig-zag ordering present within the JPEG algorithm. When the frequency terms are added to the representation of the image in this order, we have found through experimentation in the lab, that the addition of a frequency term every two to three samples provides us with excellent results. This allows for the rapid introduction of frequency terms with no loss of stability in the calculation of the inverse matrix.

4.3 Choosing the Sampling Sequence

The choice of the sequence of samples taken within the 8 by 8 pixel blocks is of vital importance to the quality of the image that is reconstructed as the algorithm progresses. Although theoretically, any sequence of sampling positions could be used with the curve fitting technique of the proceeding sections, certain sampling sequences will provide for the best development of the frequency representation as terms are introduced and refined. Specifically, since terms are being introduced in a low to high frequency manner, sampling positions should be chosen such that the accuracy of the frequency terms is refined in a similar fashion. This implies that the accuracy of the low frequency terms should be maximized first, with accuracy progressing into the higher frequencies as the number of samples increases.

In order for a minimal amount of aliasing to be introduced into the lower frequency terms, the sampling sequence should be chosen such that all samples are spaced as widely as possible during the progression of the algorithm. This fact is illustrated in figure 7, where two cases are presented. In figure 7A we have the case where two closely spaced samples returning differing intensity values are taken during the early stages of the algorithm. At this point only low frequency terms are available to fit the sample points. The fitted curve that results is illustrated within the figure. Note that although the samples are accurately fit with this curve, this representation is undesirable since it allows the reconstructed intensity values to vastly exceed the range of intensities returned by the supporting samples. The case presented in figure 7B is preferable. In this case the samples are spread as widely as possible. The fitted curve that results is again indicated within the figure. However, this

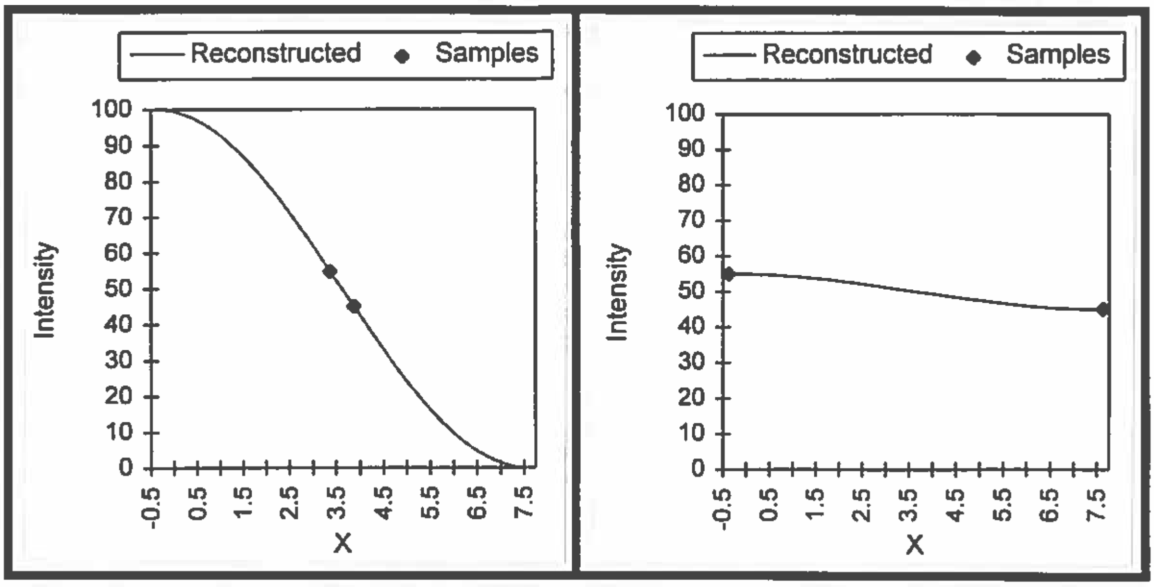

Figure 7: The results of the curve fitting technique being applied to: A) Two closely spaced samples (left). B) Two widely spaced samples (right).

time the reconstructed intensity curve is contained within the range of intensities present within the original samples. This provides for a smooth and contained transition between the sampled intensities across the entire range of the reconstructed function.

The widest spacing possible for a given number of samples is obtained when the samples are arranged in an evenly distributed regular grid. In addition, distributing samples in this manner provides for the most even coverage of the sampled area. This guarantees that the reconstructed function is adequately supported across its entire range of values. We therefore employ a method whereby sampling positions are iteratively determined such that they form regular grids of increasing resolution. This is done in a manner such that the samples are evenly distributed across the sampled area at all times.

In addition, a common sampling sequence is employed across all blocks which comprise the image. This allows for a significant reduction in the number of calculations necessary to compute the frequency domain representation. Reference to equation 22 will show that the contents of the J, and subsequently, the J^{-1} matrix which are used to compute the frequency domain coefficients, are based solely on the number and positioning of the samples given by (x_n, y_n). This implies that if all blocks employ the same progression of sample points, the matrix J^{-1} used to compute the frequency representation of one block of N samples will be the same as the matrix J^{-1} necessary to compute the frequency representation of all other blocks after N samples. Further, since frequency terms are being added at a fixed rate, the number of frequency terms present after N samples will be the same for all blocks as well.

Therefore, we iteratively construct a shared list consisting of the position of the sample to be taken and the matrix J^{-1} that is necessary to solve for the frequency representation at that point in the list. This list is then indexed into by all blocks to obtain the sampling position and matrix J^{-1} to be used. The first block to reach the end of the list then computes the location of the $(N+1)^{th}$ sample and the new matrix J^{-1} that is necessary to solve for the frequency representation of the image containing the additional sample point. In this way the number of times the matrix J^{-1} must be calculated is reduced from the total number of samples taken across the entire image, to the maximum number of samples taken within any given block, a significant reduction in cost.

4.3.1 The Quad-Tree Based Sampling Method

In order for the sampling sequence presented in the proceeding section to be obtained, a method must be defined that can iteratively construct an evenly distributed sampling sequence from positions given by a series of regular grids at increasing resolutions. The method proposed is based upon a quad-tree subdivision of the sample space.

The first four samples in this sequence are taken at the outermost four corners of the block in the order given in figure 8A. The topmost node of the quad-tree is now defined to be the region consisting of the entire 8 by 8 block. Within this region three sample

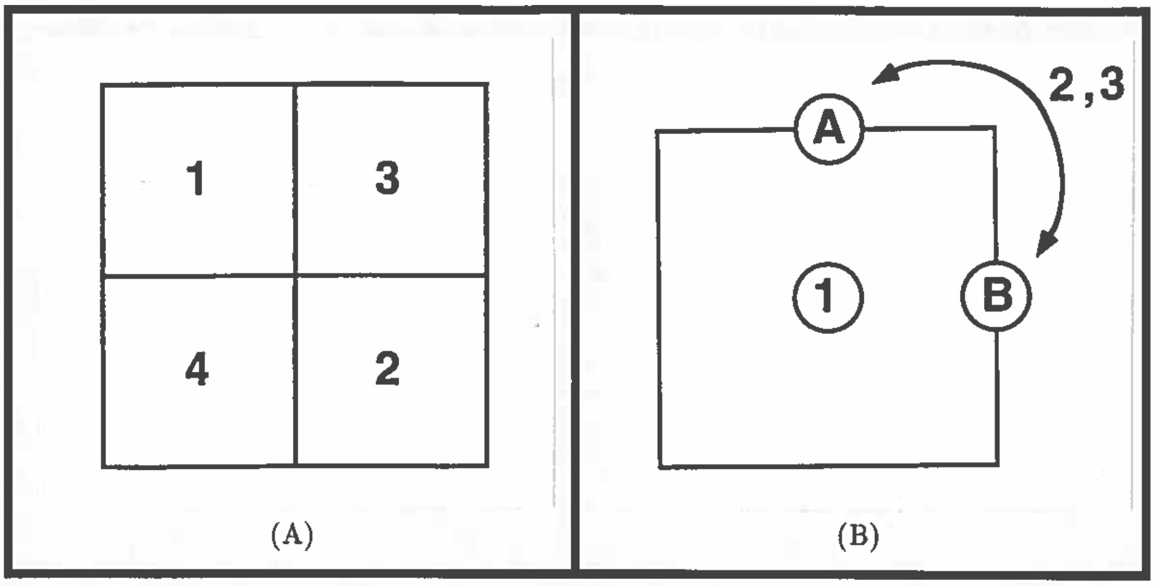

Figure 8: Ordering metrics used to drive the selection of samples within the quad-tree subdivision method. A) Sub-block ordering. B) Sample ordering within a sub-block.

positions are defined with the ordering given in figure 8B. The first sample is taken at the center of the region. A random choice is then made between position A and position B at the boundary of the region to obtain the next sample. This choice is then recorded within the node. This random choice between sampling positions is performed to evenly distribute samples in the horizontal and vertical directions.

A trick is necessary at this point. Notice that no samples are taken at the bottom or left side of a region. This is because at deeper depths of the tree, these sampling positions will be supplied by the top and right side of adjoining sub-blocks. However, at the boundary of the 8 by 8 block we have no adjoining sub-blocks present within our tree. Therefore, we simulate this with "imaginary bounding blocks". What this means is that whenever a sample is added to the list at the boundary of the 8 by 8 block, a following sample is immediately added to the list at the opposing edge of the 8 by 8 block. This simulates the addition of samples to the sequence that would be caused by the "imaginary" sub-blocks that would bound the 8 by 8 region.

We therefore, add another sample to our list on the opposing edge to the sample position chosen above. The next sample is then taken at the remaining position A or B that was not chosen originally. This sample also lies on the edge of the 8 by 8 block so again an additional sample is added. This time it is positioned at the other opposing edge.

All samples have now been taken within this node of the tree so the region is subdivided. This yields four smaller nodes each containing a region corresponding to a quadrant of the parents node's region. A breadth-first search of this quad-tree is then performed to determine the next sample positions. This method is used to evenly distribute samples across all regions of the 8 by 8 block. The ordering in which the sub-nodes are visited is again given by figure 8A. The next sample is given by taking the first sample position from sub-block 1. The following sample is given by the first sample position from sub-block 2. The process then continues, creating new nodes and recursing down the tree as necessary.

In this manner samples are iteratively taken which combine to produce the alternating Cartesian and Hexagonal sampling patterns depicted in figure 9. The first 14 sample positions given by the above algorithm are subsequently depicted in figure 10.

Note should be taken that samples which lie on the edge of an 8 by 8 block are especially advantageous to our algorithm. This result stems from the fact that such samples may be used to refine the frequency representation of all 8 by 8 blocks that they border. The corner samples are the most advantageous of all since a single sample taken at those positions may be used to refine the frequency representation of all four blocks that they border. However, the strict ordering of samples that is necessary for efficiency precludes us from immediately utilizing these samples in adjoining blocks. A sample cache is therefore employed at each block to store such boundary samples until the point is reached in the block's sampling sequence at which the sample may be utilized. This method prevents us from having to

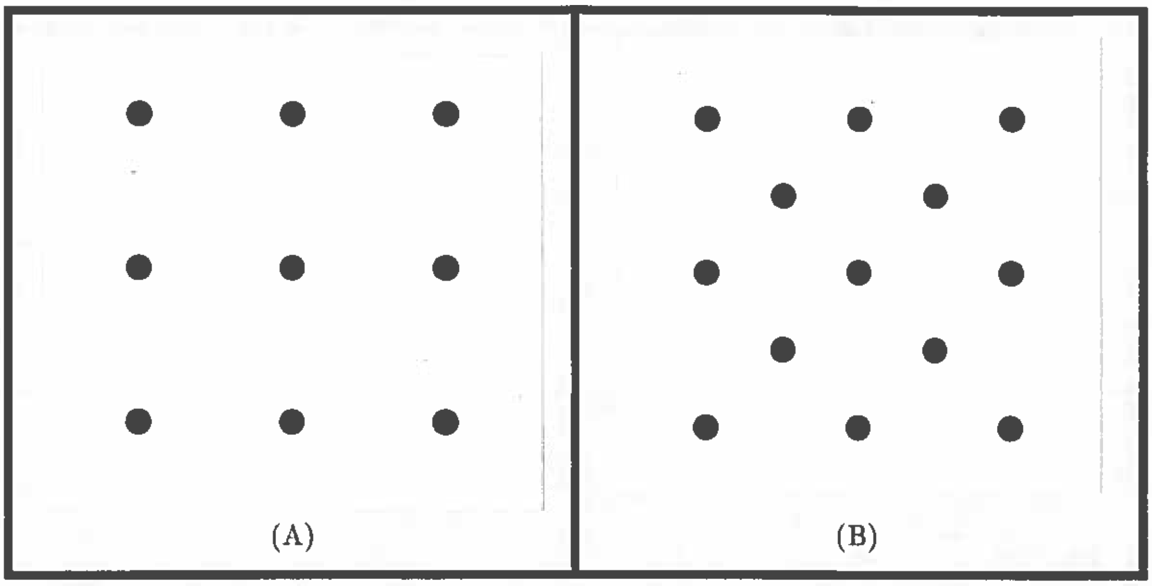

Figure 9: Sampling patterns produced at alternating positions within the sampling sequence. A) Cartesian B) Hexagonal

resample the scene at redundant locations as the algorithm progresses.

In this manner an iterative sampling sequence is produced which not only provides for the most accurate calculation of the low to high frequency terms present in the representation of the image but, additionally, allows us to exploit the redundancy present in the least squares curve fitting technique to further speed the calculation of the frequency terms.

4.4 Choosing the Next Block to Process

All 8 by 8 blocks within any given image do not generally exhibit the same amount of variation in the intensity values necessary to represent that block. Most regions of the image are generally constant or smoothly varying with only a few regions exhibiting high amounts of variation. Nyquist's theorem tells us that regions of the image with low frequency content can be represented with far fewer samples than those regions containing higher frequency content. Working in the frequency domain provides for a novel means to determine what regions require higher sampling and what regions can be adequately represented with a

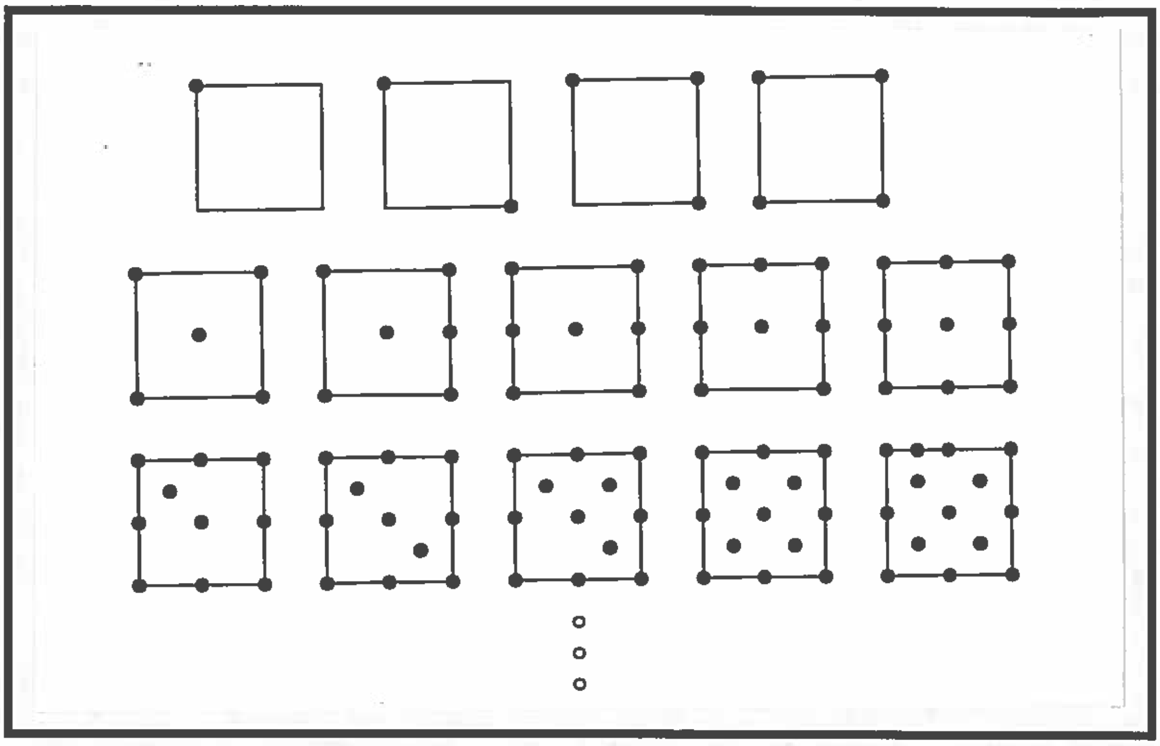

Figure 10: The initial 14 sampling positions given by the quad-tree sampling method.

limited number of samples.

Fourier theory tells us that under-sampled regions of the image will exhibit aliasing in the frequency representation of the region. This is caused by the overlapping of spectra caused by the convolution of the frequency representation of the scene with the sampling impulse train. As the sampling rate increases, the impulses forming the sample train move further apart and the overlap, or aliasing, decreases. This reduction in aliasing manifests itself as variation in the frequency spectra of the image.

Having an iteratively developing frequency representation provides for a means whereby this variation in the frequency spectra may be directly measured and used to control the sampling of the image. The underlying assumption being that blocks in which a large amount of aliasing has occurred tend to need refinement the most and, blocks in which a small amount of aliasing has occurred tend to be well defined.

In order to determine which blocks have exhibited the greatest aliasing, and therefore should be the first to receive an additional sample, a priority metric needs to be defined. The priority metric we have defined consists of two parts. First an initial uncertainty term is added to the metric representing our confidence that the underlying region has been sampled adequately enough that its aliasing estimate can be relied upon. This term dominates the equation initially but is attenuated rapidly as the sampling rate increases. This term guarantees a minimum coverage of all regions of the image. This is necessary since we do not want a region to never be sampled again merely because the first two samples returned the same intensity values. Secondly, and most importantly, a measure of the aliasing that has occurred is added to the metric. This measure is formed by the mean squared variance in the frequency representation of each of the color channels, that occurred as the result of the last sample.

These two values are summed and a gradient is formed to combine the current priority estimate with older priority estimates. These estimates are combined with decreasing weight being assigned to older priority values. This step is again necessary, as we do not want a single sample that did not alter the frequency representation to halt sampling of that block. We instead desire that the weighted average of the aliasing that is occurring over time, be used to drive the metric.

Once this priority metric has been computed a method must be employed to choose which block should be sampled next. In order to do this, all blocks within the image are placed in a sorted linked list based on their priority. The block at the head of the list is the one that is chosen to sample next. After it has been sampled and its new frequency representation has been computed, a new priority measure is computed and the block is placed back in the list and sorted based upon its new priority. This method then iterates until stopped or, a fixed aliasing threshold has been reached.

In this manner a unique form of adaptive subdivision is performed upon the image in which low frequency regions of the image will be rapidly defined and only those regions consisting of higher frequency content will exhibit the greater aliasing required to receive additional samples. Further, those areas of the image exhibiting the largest aliasing and thereby creating the most objectionable visual artifacts will be concentrated upon first, and the most heavily.

4.5 Displaying the Final Result

Once a single sample has been taken within each block of the image a functional representation of the image is obtained. This representation is then displayable at the full image resolution and will continue to be refined as the algorithm progresses. This representation can be used to continually update the display as the image resolves, providing the user with valuable early feedback.

The algorithm presented in the proceeding sections to iteratively and adaptively compute the frequency representation of an image, is fundamentally an alias driven technique. As such, there is a certain amount of aliasing present in the frequency domain representation of the image. This occurs especially in the early stages of synthesis before the algorithm has progressed to the point that the aliasing present in all blocks has been reduced below some small fixed threshold.

The algorithm therefore, must provide a means whereby this remaining aliasing may be low pass filtered and removed. The method employed should additionally be adjustable, providing for a small filter width when only a few samples have been taken and aliasing is occurring in the low frequencies, and grow larger as more samples are taken and the aliasing is limited to the higher frequency terms. Fortunately this problem is simplified by the fact that we are synthesizing directly into the frequency domain. This domain allows us to low pass filter a block of the image by simply multiplying the filter with the frequency terms. This is opposed to the more costly convolution that is necessary within the spatial domain.

We know that the accuracy of the image is proportional to the square root of the number of samples (Lee, Redner and Uselton 1994). Therefore, we designed a simple two dimensional low-pass Butterworth filter with a cutoff frequency proportional to the square root of the number of samples taken within a given block. This filter must then be multiplied by the frequency coefficients of each color channel before the frequency representation may be used to reconstruct the image for display.

The technique used by the frequency synthesis algorithm described in this report to reduce aliasing in the output image is novel in many respects. First, it allows for the direct detection of aliasing within the image. Additionally, it provides an adaptive method capable of sampling these aliasing areas at a higher rate. Finally, it provides for a method whereby the remaining aliasing may be filtered in an adaptive manner based upon the local sampling rate present within specific sub-regions of the image. This is a process that could be termed difficult at best, within the conventional spatial domain.

5 Results

The resolution fan on the left side of Figure 11 was generated using the frequency domain synthesis technique presented within this report. An average of one sample per pixel was used in its generation. Note that although a low number of samples were used, the fan is still sharp, even at its bottom-most, high frequency corner. This level of anti-aliasing

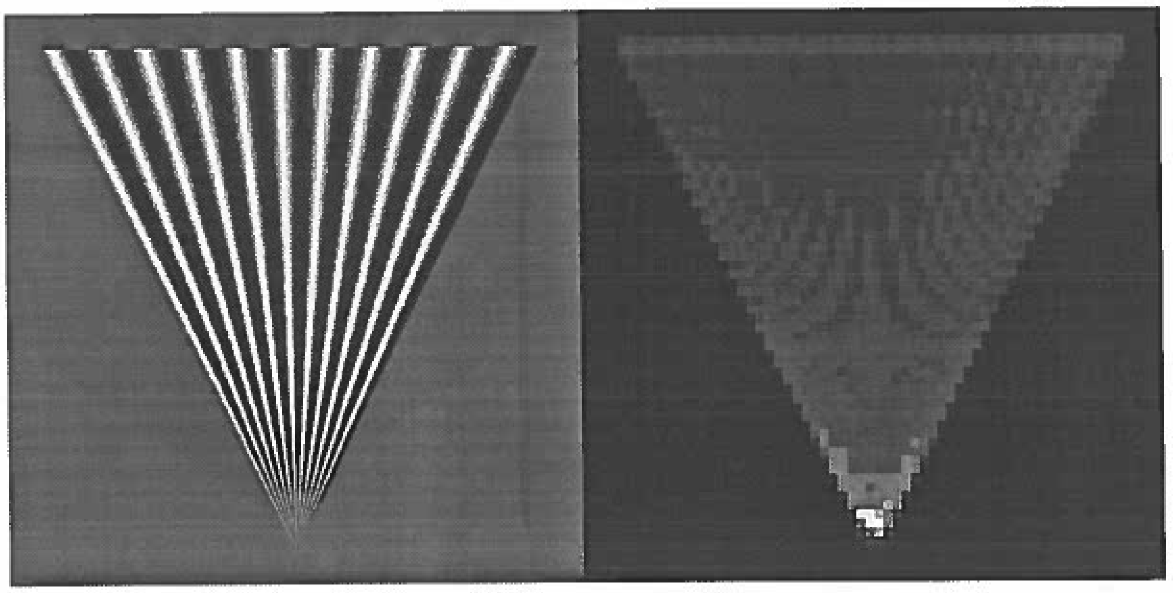

Figure 11: Resolution fan rendered using the frequency domain synthesis algorithm and the sampling density employed.

would typically require greater than 10 samples per pixel within a conventional sampling and reconstruction scheme.

The right side of Figure 11 shows the sampling density employed by the algorithm. Black regions are used to depict those areas of the image receiving the fewest samples while white regions indicate the areas receiving the highest number of samples. As we move down the fan, the spatial frequency content of the image increases, requiring a greater number of samples to accurately reconstruct the image. The sampling density image indicates that the adaptive algorithm is correctly detecting the presence of these high frequency regions and is appropriately increasing the sampling rate as we move down the fan. Conversely, the background and top of the resolution fan are relatively low frequency regions thereby requiring fewer samples. The accurate reconstruction of these low frequency areas indicates that the algorithm is able to correctly interpolate between low density samples to accurately represent slowly varying regions of the image.

Figure 12 illustrates a series of images produced during the iterative progression of the

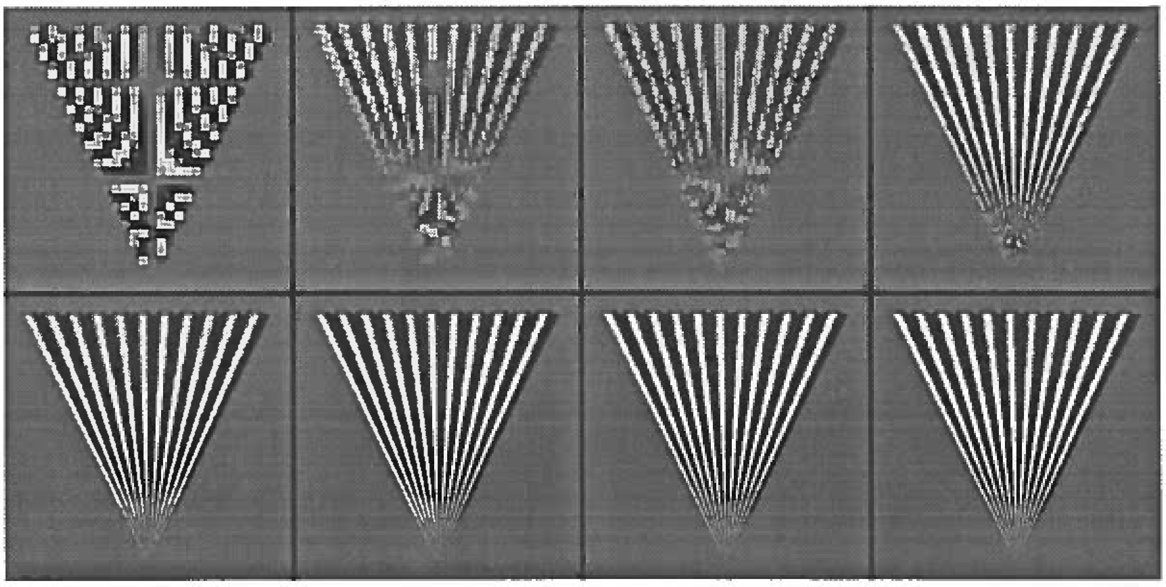

Figure 12: Iterative generation of the resolution fan. Images are depicted using and average of $\frac{1}{64}$, $\frac{1}{32}$, $\frac{1}{16}$ and $\frac{1}{8}$ (top row) and $\frac{1}{4}$, $\frac{1}{2}$, 1 and 2 (bottom row) samples per pixel respectively.

algorithm as it is being used to generate the resolution fan contained within Figure 11. The images are organized in a left to right, top to bottom order. Each successive image was generated using twice the number of samples present in the previous image. The initial image was generated using an average of $(\frac{1}{64})^{th}$ of a sample per pixel, or one sample per 8 by 8 block. The successive images were generated using an average of $\frac{1}{32}$, $\frac{1}{16}$, $\frac{1}{8}$, $\frac{1}{4}$, $\frac{1}{2}$, 1 and 2 samples per pixel respectively. As the algorithm progresses, we see that the low frequency variations at the top of the fan are resolved early, with accuracy extending down the fan to the higher frequency regions as the sampling rate increases. This illustrates the algorithm's ability to correctly reconstruct low to high frequency variations within the image as successively more frequency terms are added to the representation.

Figure 13 shows the result of the frequency domain synthesis technique being applied to a more realistic and complex scene. The chess board image on the left was generated using an average of one sample per pixel and the image on the right indicates the sampling density employed. Note that the high frequency variations between the squares on the board

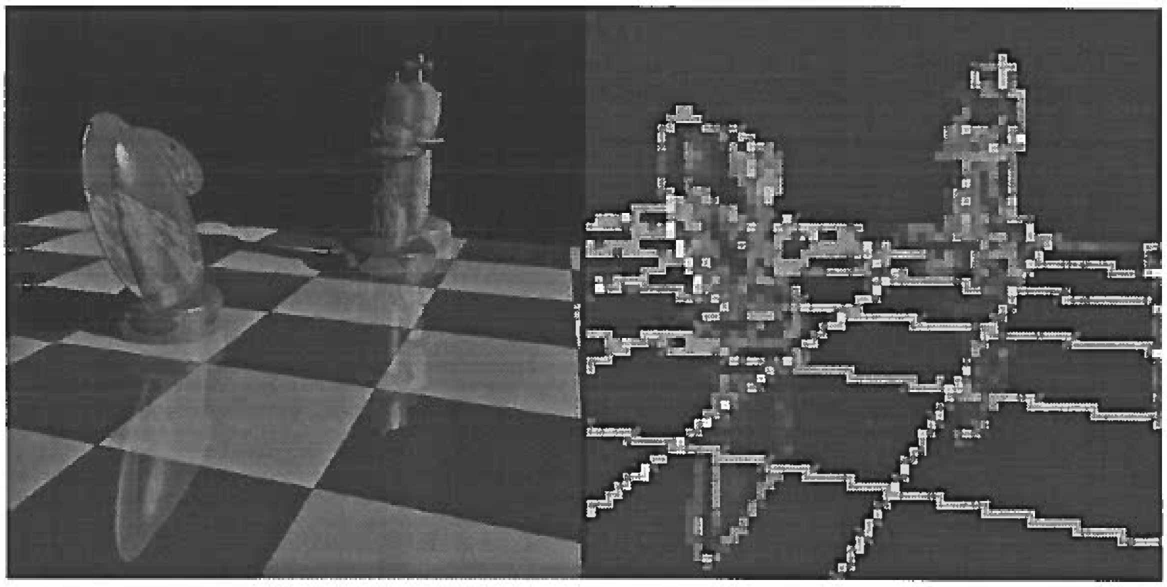
Figure 13: Chess board rendered using the frequency domain synthesis algorithm and the sampling density employed.

and within the chess pieces have been correctly detected and sampled at a higher rate. Additionally, observe that these regions have been sharply anti-aliased even though a low number of samples were used to generate the image. This illustrates that the combination of edge detection, interpolation and filtering that is utilized within the method is able to correctly resolve high frequency variations while minimizing the total number of samples taken within the image.

The progression of images generated as the chess board scene is being rendered is depicted in Figure 14. The number of samples taken for each image in the sequence is the same as for Figure 12. During the initial stages of the algorithm, low density samples are evenly distributed across the image plane. This is a result of the uncertainty term's initial domination of the adaptive metric. This even distribution of samples assures adequate coverage of the sample space and allows a low frequency representation of the image to be generated. Note that while severely distorted, basic image content is still discernible within the first few images of this low frequency representation, providing the user with valuable

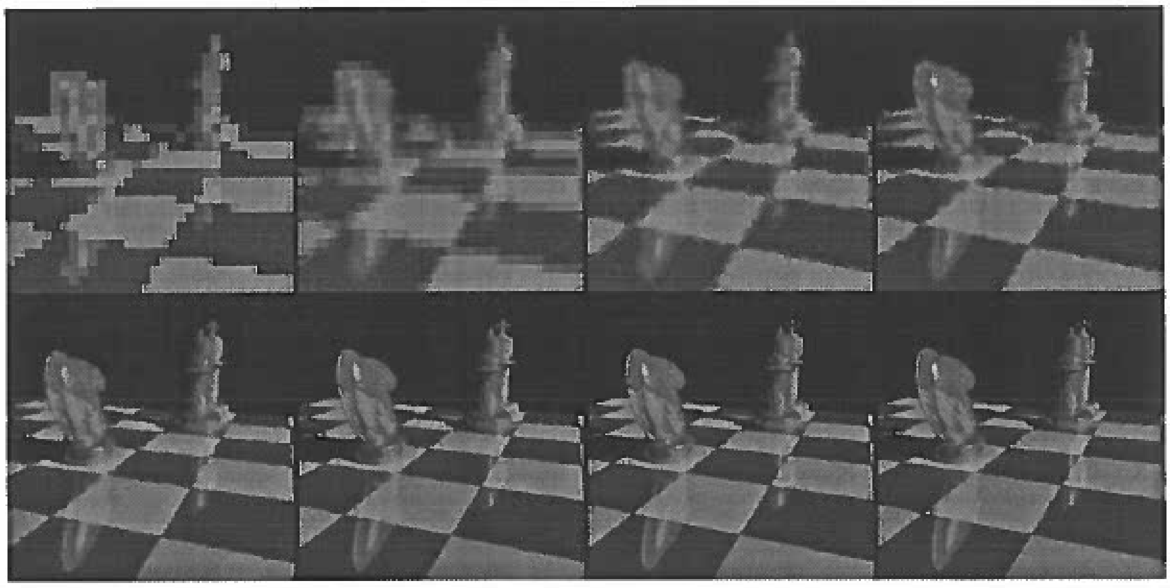

Figure 14: Iterative generation of the chess board. Image are depicted using and average of $\frac{1}{64}$, $\frac{1}{32}$, $\frac{1}{16}$ and $\frac{1}{8}$ (top row) and $\frac{1}{4}$, $\frac{1}{2}$, 1 and 2 (bottom row) samples per pixel respectively.

early feedback. As the algorithm progresses, the aliasing estimator increasingly dominates the adaptive metric. This focuses the sampling on those regions of the image wherein the most severe distortions are occurring, allowing a more accurate higher frequency representation to be obtained, without the expense of wasted samples within the well defined portions of the image. This allows the algorithm to rapidly remove the most objectionable artifacts from the image, as is visible within the later images of Figure 14.

One can note the similarities of the reconstruction process with the addition of frequency terms within the JPEG algorithm by comparing the results of Figure 14 with those of Figure 2. Observe that within Figure 14 the most severe distortions are removed during the early stages of the algorithm. This is analogous to the addition of the first few frequency terms within the JPEG representation, and illustrates the greater magnitude and visual importance of the low frequency information. Additionally, note that while the sampling rate is increasing dramatically within the latter images of Figure 14, visually it is extremely difficult to distinguish between the last frames in the sequence. This further illustrates the declining magnitude and perceptual importance of the high frequency components within the Discrete Cosine Transform representation of an image. This fact allows the new image synthesis technique to produce images of a superior visual quality with the introduction of relatively few frequency terms for most regions of the image.

6 Conclusion

A new technique has been developed that is capable of iteratively synthesizing computer generated imagery directly into the frequency domain given by the Discrete Cosine Transform. This domain allows the algorithm to directly exploit the relative importance of image frequency information, in order to generate the most visually significant components of the image first and progress to the less significant components as time and resources permits.

For this purpose, an iterative and adaptive least squares technique has been described that allows a functional representation of an image to be generated, in a low to high frequency manner. This allows the algorithm to resolve the most important, low frequency components of the image before the less important, high frequency components. As seen within Figure 12 and Figure 14, this technique is capable of providing early feedback to the user, with an excellent early and intermediate representation of the image obtained at very low sampling densities.

Additionally, the low frequency energy prevalence of most images and the energy packing efficiency of the Discrete Cosine Transform has been shown to allow the majority of an image's content to be contained within the first few frequency terms. This may be seen upon inspection of Figure 2 and Figure 3. This fact implies that the algorithm will be able to rapidly define most regions of the image with the introduction of relatively few frequency terms.

Further, it has been shown that variations in the frequency domain representation of an image may be used to estimate the amount of aliasing occurring within regions of the image. This estimate is then used to adaptively control the sampling of the scene, such that the most objectionable image artifacts are removed first. In this manner the sampling density is varied across the image in order to adaptively meet the Nyquist criteria for all regions within the image.

The sampling, reconstruction and filtering techniques employed within this report are able to avoid the majority of downfalls present within previous work of this sort. This is due to the fact that the frequency representation is directly generated for an image. This representation may then be directly filtered, without the need for expensive convolution, and re-sampled to generate the final image representation. This technique avoids the approximations that were performed during the reconstruction and filtering stages of other prior work.

Additionally, the noise present in the output image of previous techniques can be greatly attenuated through the use of the methods described in this report. This is the result of the band-limited reconstruction and least squares algorithm that is employed. This technique seeks to minimize the error between the sample values and the reconstructed intensity function. In this manner, a smooth curve is fit through potentially noisy sample values, resulting in a smoothly filtered output image that obviates the need for expensive post-filtering processes as described in Lee and Redner (1990) and Rushmeir and Ward (1994).

Finally, the techniques presented within this report can be seen as a first step to an algorithm that is capable of synthesizing computer generated imagery directly into a compressed format. The format into which this algorithm synthesizes images is consistent with the first stage of the JPEG compression process, thereby reducing the number of steps necessary to JPEG compress the resulting image. But, perhaps more important, is the fact that the algorithm expends energy and iteratively produces images that are directly related to the compressed representation of the image at successively higher quality levels. This eliminates the wasted effort involved in computing large amounts of non-visually significant information which could potentially be discarded by a subsequent conversion to a compressed format.

It is this author's belief that a large amount of unexploited synergy exists between the fields of computer graphics, image compression and vision. This work represents an initial step in that direction. This is a topic which I believe holds strong promise for the future. It is further, a rich and rewarding area for future research, with many avenues yet unexplored. It is my strong hope to continue along this course of intriguing research.

A Survey of Image Compression Algorithms

A.1 Introduction

Image compression and data compression in general, is an attempt to reduce the number of bits necessary to represent a piece of information. There are numerous algorithms that seek to do this in many different ways. The most prevalent general compression schemes do this through either a run length encoding, keyword substitution or substitutional compression scheme. The still image compression schemes generally provide two types of compression lossless and lossy. The lossless methods usually employ keyword substitution or some type of predictive encoder. The lossy algorithms however, have received the most attention lately and come in a number of forms. The most prevalent of which are transform based compression such as methods that use the Discrete Cosine Transform or wavelet transform. Recently however, schemes like subband image coding, vector quantization and fractal transforms have been receiving growing attention. Moving image compression schemes generally use the same algorithms as still image compression with the addition of extra temporal compression that is achieved by capturing the inter-frame redundancy that is prevalent in a video signal.

A.2 General Compression Schemes

General compression schemes were originally intended for use primarily as a means of entropy encoding general data, predominantly text. They are therefore not specifically tuned to the needs of image compression. However, almost all image compression schemes use some type of general entropy encoding mechanism somewhere in the process. General compression schemes will therefore be covered here to provide a groundwork for future discussions of image encoding schemes.

A.2.1 Run-length Encoding

Run-length encoding is one of the first data compression schemes and perhaps the least complicated. It is intended for use with data that contains a large number of repeated characters or, where a simplistic scheme is necessary. Usually the data is either redundant to start with or, it is preprocessed in some way to translate it into a more redundant form. Image encoding is an example of this, since neighboring pixels tend contain similar intensity values. If we only encode the change in the value of the pixel as we move across a scan line, the data tends to be filled with a large number of zeros and other small numbers. This data is then good for a run-length encoding scheme.

The basic idea behind run-length encoding is to shorten the data by encoding runs of the same piece of data, as the number of repetitions of the data followed by the data itself. An example of a simple run-length encoding scheme is the Apple Macintosh PackBits scheme. This scheme goes through the data and looks for repeated bytes. For non-repeated bytes, it stores a byte n that is between 0 and 127 inclusive that says the next $n + 1$ bytes are non-repeating. It then stores the non-repeated string. For repeated bytes it stores a number n between -1 and -127 inclusive that says the next byte is repeated $-n + 1$ times in the original data. It then stores the repeated byte. Compressed and uncompressed runs of length greater than 128 bytes are stored as multiple instances of the above. In general, it is best to encode a 2-byte repeat run as a replicate run except when preceded and followed by a literal run. In that case, is best to merge the three runs into one literal run. 3-byte repeat runs are always encoded as replicate runs. The decompression routine then looks like:

```
Loop until you get the number of bytes you are expecting:
  Read the next source byte into n;
  If n is between 0 and 127 inclusive
    Copy the next n+1 bytes literally;
  Else if n is between -1 and -127 inclusive
    Copy the next byte -n+1 times;
  Else if n is -128 nop;
Endloop
```

Run-length encoding is used in the Apple scheme above, the Utah Raster toolkit, it is an option in the TIFF file format and is used in a number of other schemes. It does not generally provide state of the art compression, except for highly repetitive files, but it is perhaps the simplest method to implement and therefore fairly widely used.

A.2.2 Statistically Based Code Word Substitution

Statistically based code word substitution schemes achieve compression by observing the frequency of occurrence of the symbols in the data and, by representing the most frequently occurring symbols using the smallest keywords and, the less frequently occurring symbols using longer keywords. Statistically based compression therefore, achieves its best results when used with data with a large, relatively fixed frequency distribution. The actual compression routines generally determine this probability by either doing a real-time frequency analysis of the entire data set before compressing the data (this requires that a code book be sent with the compressed data stream in order to decompress the data), by using a prepared code book of symbols based on the normal frequency of the symbols to be represented or, by using an adaptive model. An adaptive model starts with a fixed set of frequencies and adapts them as the data to be compressed is processed. The latest compressors however, can do even better than that by utilizing the probability of the inter-symbol dependencies as well, in their probability predictions. Examples of this are Dynamic Markov Coding (DMC) and Prediction by Partial Matching (PPM) techniques. Once the probability has been determined it is then passed to one of the compression routines such as Huffman encoding, Shannon-Fano encoding or, the new Arithmetic encoding.

Huffman Encoding Huffman encoding has been around for quite a while and was long thought to be the optimal compression scheme however, in actuality, it is only optimal for data with symbols whose frequency of occurrence are integral powers of 1/2. Huffman encoding works by:

1. Rank all symbols in probability of occurrence.
2. Successively combine the two symbols with the lowest probability to form a new composite symbol. This step is repeated to build a binary tree where each node is the probability of all nodes beneath it, and the leaf nodes represent the actual symbols.
3. To determine the keywords to substitute for each symbol, trace a path from the root node to the leaves, at each branch of the tree assign one branch a binary 0 and the other branch a binary 1. The keyword is then the combination of 1's and 0's encountered as we traverse from the root to the leaf node containing the symbol.

Shannon-Fano Encoding Shannon-Fano encoding is similar in approach to Huffman encoding, it just goes about it in a different manner. Huffman encoding generates symbols in a bottom-up approach while Shannon-Fano utilizes a top-down method. Shannon-Fano encoding works as follows:

1. Divide the set of symbols into two equal or almost equal subsets based on the probability of occurrence of characters in each subset. The first subset is assigned a binary 0 and the second a binary 1. This step is repeated until all subsets have a single element.
2. The keyword is then determined by combining the 0's and 1's of ensuing subsets until the subset containing exactly the symbol is reached.

Arithmetic Encoding Although Huffman and Shannon-Fano encoding are often touted as the optimal compression techniques, in actuality this is only the case when the symbols in the data occur in probabilities that are integral powers of $\frac{1}{2}$. Arithmetic encoding however, lifts this restriction. Arithmetic encoding works by representing the entire data stream as a single (often very long) fractional number between 0 and 1. This number is chosen such that it is within the range of the probability of the data stream's occurrence. This is accomplished as follows. Initially the probability of occurrence of each symbol is mapped onto a section of the range from 0 to 1 that is equal in size to the probability of its occurrence. So if we had two symbols possible; X and Y with probabilities $\frac{2}{3}$ and $\frac{1}{3}$ respectively. Then X could take the range of 0 - $\frac{2}{3}$ and Y take the range $\frac{2}{3}$ - 1. Then a symbol, say X is read from the data stream. The probability of the data stream is then reduced to that symbol's range of probability. In this case the range 0 - $\frac{2}{3}$. Then the symbol probabilities are mapped onto this new range. In our example this would mean that the probability of receiving another X (or of the data stream being XX) would take the range 0 - $\frac{4}{9}$ and the probability of receiving a Y (or of the data stream being XY) would take the range $\frac{4}{9}$ - $\frac{2}{3}$. This process is repeated for the entire data stream. In the end we have the range of possibility of receiving exactly the data stream. The fractional number that can be represented with the fewest bits within this range is then outputted as the compressed string.

A.2.3 Substitutional Compressors

Substitutional compressors are based on algorithms put forward by Abraham Lempel and Jacob Ziv in 1977 and 1978. In practice, substitutional compressors work very well and are the basis for numerous commercial compression products. The basic idea behind substitutional compressors is to replace repeated occurrences of pieces of data with a reference to previous occurrences of that piece of data. There are two main classes of LZ compressors, the 1977 version termed LZ77 which uses a sliding window scheme and the 1978 version, LZ78 which stores repeated phrases in a dictionary for future lookup.

1977 Limpel Ziv (LZ77) family of encoders The LZ77 family of encoders work by keeping track of the last n bytes of data. Then when a phrase is encountered that has already been seen, it is replaced by a pair of values corresponding to the position of the phrase in the previous bytes of data and, the length of the phrase. In effect this algorithm is sliding a window over the data and replacing duplicate entries in the window with pointers to previous entries. An algorithm based on the most common of these schemes LZSS by James Storer and Thomas Szymanski developed in 1982 maintains a lookahead buffer, wherein it tries to match the longest possible phrase starting at the beginning of the buffer with data in the previously seen sliding window buffer. The algorithm is as follows:

```
while ( LookAheadBuffer not empty )
 {
  get a pointer ( position, length ) to the longest match in the
     window for the lookahead buffer;
  if ( length > MAXIMUM_MATCH_LENGTH )
    {
    output a ( position, length ) pair;
    shift the window length characters along;
    }
  else
    {
    output the first character in the lookahead buffer;
    shift the window 1 character along;
  }
}
```

Decompression in then just a matter of copying the data until a (position, length) pair is encountered and then copying "length" bytes from the position indicated in the window, which is maintained by the decompressor as well.

1978 Limpel Ziv (LZ78) family of encoders The LZ78 family of encoders work by entering phrases into a dictionary and, then when a repeat occurrence of that particular phrase is encountered, the dictionary index is outputted instead of the actual phrase. The most popular of these schemes is Terry Welch's LZW scheme, a variant of which, LZC, is used in the UNIX compress program, as well as other places.

The LZW scheme starts with a 4K dictionary, in which, entries 0 - 255 are initialized to refer to individual bytes and, entries 256 - 4095 are used to refer to substrings which are built as the algorithm goes along. New substrings are generated when a string is encountered which is one character longer than an existing entry in the dictionary. The output data stream is then just the number of the entry in the dictionary that corresponds to the string parsed (Note: When just a single byte matches the dictionary the value outputted is the actual value of that byte). This algorithm is as follows:

```
Initialize entries 0 - 255 to be the byte corresponding to their
  entry number;
set w = NIL;
while ( more data to compress )
  {
  read a character K;
  if ( wK exists in the dictionary )
    w = wK;
  else
    {
    output the code for w;
    add wK to the string table;
    w = K;
    }
  }
```

The nicest feature of this type of compression is that the dictionary need not be sent to the decompressor. This is the result of the fact that the contents of the dictionary is inherent in the compressed string, so the decoder is able to automatically build the dictionary as it decompresses the string. Later variations on the LZ78 method add such things as a LRU scheme to their dictionary as well as modifying the length and method of building dictionary entries.

A.3 Image Compression Schemes

A.3.1 CCITT Facsimile Standards

The CCITT Facsimile standards are largely concerned with the lossless representation of bi-level or binary (one bit/pixel) images for transmission (to FAX machines). These standards contain a series of compression algorithms such as Group 3 and Group 4 compression specified in the T.4 and the T.6 draft of the standard, as well as the new JBIG compression standard.

T.4 (Group 3) and T.6 (Group 4) Compression Standards The Group 3 and Group 4 compression standards offer two options in their encoding scheme, one-dimensional and two-dimensional coding. One-dimensional coding is a simple statistically based encoding mechanism. It represents each run of black or white in the scan line as a codeword or group of codewords. There are codewords or combinations of codewords available for all lengths of runs. These codewords are listed in the standard and are based on the statistical frequency of runs of different lengths. There are different codebooks for black and white codewords. The basic algorithm then starts by assuming the scan line begins with a white run. If it actually begins with a black run, the keyword for a white run-length of zero is outputted. The length of the runs in a scan line are then determined, as well as whether they are black or white runs. The corresponding codewords for the line are then looked up in the codebook and outputted. An End-of-Line codeword is outputted at the end of each line to minimize errors due to a noisy communication medium. This process is repeated for the entire data.

Two-dimensional encoding is an optional extension to one-dimensional encoding. It allows a fixed number of two-dimensionally encoded strips to be encoded between one-dimensional scan lines. Two-dimensional encoding is basically a predictive scheme which utilizes the positions of three changing elements (black to white or, white to black) on the current scan line and two on the line above it. These elements are the current element being encoded, the following two changing elements on the same scan line and, the first two changing elements on the line above and to the right of the current element. A series of codewords are then output from the codebook that matches (by a couple of different mechanisms) the relative color and positions of these five transition, position pairs.

JBIG Compression Standard The JBIG algorithm was developed by the Joint Bi-level Image Experts Group of ISO, IEC and CCITT to replace the existing outdated Group 3 and Group 4 facsimile compression standards. The JBIG algorithm is intended for bi-level images but, may be extended to gray-scale images by encoding the bit-planes separately. This tends to work well up to 6 - 8 bits per pixel at which point it is better to encode the image using JPEG's lossless mode. The algorithm provides for both a progressive and a sequential coding mode. In the progressive mode the image is transferred at increasing resolution so that the receiving machine can gradually build up the image. The sequential mode sends the image directly.

The algorithm utilizes a predictive scheme that models the redundancy in the image as the correlation of the pixel currently being coded with a set of nearby pixels called the template. An example template might be the two pixels before this one on the same line and the five pixels centered above this one on the previous line. Templates may only be chosen from pixels that are above or to the left of the current pixel (i.e., they have already been seen from the scanner). The current pixel is then encoded based on the state of the current pixel and the pixels in its template. The Arithmetic coder and probability estimator used are actually IBM's patented Q-Coder (Pennebaker, Mitchell, Langdon Jr. and Arps 1988 and Pennebaker and Mitchell 1988), which uses adaptable probabilities as well as adaptive templates to improve the compression.

A.3.2 Frequency Based Transforms

Frequency based transforms like the Discrete Cosine Transform and the wavelet transforms, while not strictly compression mechanisms, do provide for a more easily compressed form. These transforms convert the input image from its original gray-scale image domain into a frequency domain. In this frequency domain the data may be represented more compactly due to the fact that a large percentage of images tend to have constant areas or, at least areas that vary gradually. This means that the majority of the image energy will be concentrated in the low frequency coefficients in the frequency domain and, the higher frequency coefficients will be mostly zero. When this transform is then combined with a scalar quantization of coefficients and an integer rounding process, the image can then be stored very compactly, generally by a representation consisting of only a few of the lower frequency coefficients.

The Discrete Cosine Transform The Discrete Cosine Transform works by decomposing the original image into a series of harmonic cosine waves, in a manner similar to the Discrete Fourier Transform. The original image can then be exactly reproduced by the sum of these cosine waves. The Discrete Cosine Transform then provides us with a formula whereby we may determine the coefficients of these harmonically related cosine waves which will sum to reproduce the original image. This transform also has the nice property than an N by N image may be exactly reproduced by N by N harmonically related cosine waves multiplied by their coefficients given by the transform. The image may therefore be completely represented by a N by N array of coefficients, the majority of which will tend to 0 in the higher frequencies.

Wavelet Transforms Wavelet transforms are relatively new to the field of image compression and a large amount of research is currently under way to determine the breadth of their impact. While wavelet transforms have not yet found their way into any of the major compression standards, they seem to have numerous properties which make them attractive for use in image compression.

Wavelet transforms have used the sinc function [$sin(x)/x$] as their basis function, as opposed to the cosine function used in the Discrete Cosine Transform. The sinc function seems to handle edges better than the cosine function. When a sharp edge is encountered by the Discrete Cosine Transform, large high frequency elements occur which greatly reduce the possible image compression. The wavelet transform however, has been likened to the edge detection function present in the human retina, and therefore handles these situations much better.

The wavelet transform uses as its basis, the fact that Shannon proved in the forties; that one can discretely sample a continuous function (or a high resolution discrete image) utilizing a sinc aperture at increasing resolution to produce a set of signals that can be combined back together to produce the original image. If these sinc functions are then differenced from the next highest frequency sinc function a set of images result, which show the edges at different resolutions. The combined set of data that results is then the equivalent of a low resolution image and a sum of increasingly higher resolution difference images that represent the edges at different resolutions. The coefficients of these sinc functions are then the wavelet transform.

A.3.3 JPEG Compression Standard

The JPEG compression standard is the result of a push by ISO and CCITT to develop an international standard for the compressed representation of still images. Previous to this standard, and still today to a large extent, images have been compressed by a large number of ad hoc, in-house compression schemes. ISO believes that producing a standard representation for the compressed form of images will result in more frequent and easier sharing of information, as well as allowing for greater interoperability of consumer products. This is especially important as digital imaging and multimedia communication becomes more prevalent.

The JPEG compression scheme contains both a lossy and a lossless mode. The lossy mode works by breaking the image into 8 by 8 pixel blocks. The value at each pixel is represented by the gray-scale value of the pixel. Color images may be represented by breaking each 8 by 8 block into numerous blocks for each color channel, for example with separate blocks for red, green and blue. The values of the 8 by 8 block are then transformed into the frequency domain by means of the Discrete Cosine Transform. The new values take the form of an 8 by 8 block of frequency coefficients where each value is the amplitude of a harmonic cosine wave. The sum of which may be used to represent the original signal. These values are organized with the DC value in the upper left corner of the block and increase in frequency to the highest frequency values in the lower right corner. These coefficients are then quantized by means of a scalar quantization table specified by the application. This step allows the algorithm to achieve further compression by discarding information that is not visually significant by assigning more bits to the representation of the lower frequency elements. The DC values are then handled separately from the AC values. The DC values are encoded as the difference from the DC term in the previous 8 by 8 block. This is known as Differential Pulse Code Modulation (DPCM). This is done to exploit the strong correlation between the average color of adjacent 8 by 8 blocks.

The AC values are then read in a zigzag order going from the upper left hand corner to the lower right. This helps to facilitate the next entropy encoding step, by placing the low frequency coefficients, which are more likely to be non-zero before the high frequency coefficients, which are more likely to be zero. This output is then encoded using either Huffman or Arithmetic encoding. An example of how this would be done, would be to first transform the data into intermediate symbols as follows. The DC values would be differentially encoded as above into the representative pair (size)(amplitude), where size is the number of bits used to represent the variable length integer (VLI) of the amplitude and, the amplitude is value of the difference in the DC coefficient from the previous 8 by 8 block. The non-zero AC values are then transformed into the representation (run length of 0's, size)(amplitude) where the run length term contains the number of 0's before the non-zero term in zigzag order from the 8 by 8 frequency representation, the size is again the number of bits used to represent the amplitude VLI and, the amplitude is the size of the non-zero frequency coefficient. A special end of block (EOB) symbol is used to represent the end of the 8 by 8 block's representation. Thus the intermediate representation for an 8 by 8 block looks like: (DC size)(DC amplitude VLI) (AC run length of 0's, size)(AC amplitude VLI) (AC run length of 0's, size)(AC amplitude VLI) ... EOB. The (DC size), (AC run length of 0's, size) and EOB terms are then represented using a predefined Huffman or Arithmetic variable length code (VLC) table. Thus, the final output data stream becomes: (DC VLC) (DC VLI) (AC VLC) (AC VLI) (AC VLC) (AC VLI) ... (EOB VLC).

The JPEG lossless mode is a predictive compression algorithm which was added to the standard for those who required a mode which could exactly reproduce the original image. The image is first transformed by predicting each pixel on the basis of the pixels above and to the left of it in the image. Seven different prediction algorithms are specified in the standard. The predicted value is then subtracted from the actual value and, this difference is then passed to an entropy encoder, which is the same as the one described for the DC coefficients in the lossy algorithm. The compressed variable length codes and integers are then output.

A.3.4 Subband Image Coding

Subband image coding provides an alternative method for the sampling of images, which can provide a more efficiently compressible form. The basic idea is to translate the image into the 2-dimensional frequency domain. Therein the frequency range is subdivided, for example into 4 subbands. These subbands are then further decimated by a factor of 2 in both dimensions. This subband is effectively the equivalent of the original range sampled at a lower rate. These subbands are then split again and the process repeats. This signal can then be decoded by up sampling the signal by 2 in both dimensions, band-pass filtering it to remove aliased copies and, then summing all of the subbands.

This subband representation is then a more efficient representation for compression. It is then typically compressed by using a predictive compression scheme on the individual subbands. Subband image coding also has the nice property that since each subband is transmitted separately, it is possible to progressively build the image at the decoder as increasingly high resolution subbands are transmitted.

A.3.5 Vector Quantization

Shannon's rate distortion theory showed that better compression performance can always be achieved by coding vectors instead of scalars (i.e., blocks of an image versus single pixel values). Vector Quantization is an implementation of this.

The basic algorithm utilizes a codebook of image blocks, which are then matched to sections of the image to be compressed and, the indexes of the image blocks in the codebook which best match sub-sections of the original image are then output. The image blocks in the codebook are matched to equivalent size sub-blocks in the original image, by finding the codebook block match which has the least error, usually measured by least-squared error or by the Itakura-Saito distortion measure.

The difficult part of Vector Quantization lies in determining a good codebook. The general mechanism of which is to define a training set of images which are then passed through the compression scheme. This implies that they are subdivided and matched with an initial codebook. The initial codebook may be produced by a number of mechanism, one of which is to simply use a number of random sub-blocks from the training data. If the average distortion produced by the compression scheme is less than some tolerance then quit. Otherwise, replace the old codewords (image sub-blocks) with a new codeword which reduces the mean error of all blocks of the training data which mapped to this codeword. This process then repeats until all codewords are within tolerance.

The are also more advanced forms of Vector Quantization which instead of using fixed size codewords, use adaptive means to generate codewords of varying block sizes. Organization of the codebooks into tree structures is also used to minimize the time necessary to match portions of the image with their best codeword.

A.3.6 Fractal Compression

Fractal compression is one of the newest image compression techniques and is rumored to provide one of the best compression ratios. Presently fractal compression has been limited by its newness and by the fact that the compression algorithm takes a long time to run, although the increasing speed of computers should reduce this restriction in the future. The decompression routine however runs very fast, making it an advantageous algorithm for compress once, decompress many, types of applications. Fractal compression also has an advantage in that it represents detail at every scale. This does not mean that a compressed image of a human face could be repetitively enlarged to show cells and eventually atoms. However, this does mean that fractal images decompressed on a larger scale will not suffer from increased pixelization as would a normally enlarged image.

Fractal compression works by determining self-similarity in images and representing the image as a transform based on these self-similar regions. The basic fractal compression algorithm attempts to find a series of maps for all pieces of the image to larger, similar pieces of the image. A map defines the rotation or flipping of the larger image to get it to closely match the smaller image. A map also defines the variation that is necessary in the contrast and the brightness to go from the larger piece, to close to the brightness and contrast in the smaller piece. I am assuming gray-scale images here, which is easily expandable to color images by considering each channel separately. These maps are then the only output of the compression routines, encoded in some reasonable manner.

In practice fractal compression can be done by arranging the image into a quad-tree representation, where the image is broken into 4 quadrants, each of which is broken into four more quadrants and, so on until we reach a reasonable starting depth (breaking the image down to a size of 8 by 8 for an original image size of 256 by 256 is probably reasonable). Then for each square in the quad-tree, attempt to cover it with a square that is larger. This is done by shrinking the larger square to the smaller square's size, usually by averaging pixels or by some similar means, and then trying to rotate the square by 90 degrees or by flipping it over and rotating it (8 positions are possible). Then determine the variation in the contrast and brightness using a least squares regression. Finally, if the error between the mapping and the original square is within a specified tolerance accept the mapping otherwise, sub-divide the square further and repeat the process.

Decompression then occurs by starting with a blank (or really any image) and successively applying the maps, until the desired resolution is acquired. This works by mapping a larger section of the picture onto a smaller one, with varied contrast and brightness. The repetition of which gradually evolves into the original image.

A.4 Moving Image Compression Schemes

A.4.1 Px64 Video Coding Standard

The Px64 coding standard is actually CCITT Recommendation H.261, Video Codec for Audiovisual Services at px64 kbit/s. This recommendation attempts to define a standard for video telephony and videoconferencing. The specification is based on services for ISDN at px64 bit rates, where p = 1 - 30. This will cover the entire ISDN bandwidth. P values of 1 and 2 are recommended for video telephony and $p > 5$ for videoconferencing. The specification is at 30 frames per second with the option of dropping frames if the bit rate gets too high. The recommendation specifies two formats; Common Intermediate Format (CIF) and Quarter-CIF (QCIF). Where everyone implementing the standard must support QCIF and support of CIF is optional. CIF is specified at a resolution of 352 by 288 for the Luminance (Y) channel and a resolution of 176 by 144 for the chrominance (CB and CR) channels. This resolution format is known as SIF and is based on the CCIR-601 digital television standard (used by professional video equipment), decimated by 2:1 in the horizontal direction, 2:1 in the time direction and an additional 2:1 in the chrominance vertical direction. QCIF is specified at half of CIF's resolution in both dimensions.

The compression works in a manner similar to the JPEG compression standard with the addition of temporal compression based on interframe redundancy. The standard supports two types of coding, intraframe and interframe. Intraframe coding removes the spatial redundancy only, in a manner similar to the 8 by 8 block JPEG encoding. Intraframes are used for the first frame and, after a change of scene. Interframe coding removes the temporal redundancy by using a predictive motion estimation scheme. This is done for the luminance blocks only, chrominance blocks are not encoded using interframe techniques. The prediction scheme works by comparing the 8 by 8 luminance block with the 8 by 8 luminance blocks in the neighborhood of the block in the previous frame. A motion vector is then determined for the prediction and only the difference between the prediction and the actual block is handed to the spatial reduction scheme (similar to the JPEG algorithm). If the difference between the prediction and the actual data is within a certain threshold, no data is sent. The quantization phase of the spatial reduction scheme is also adjustable based on the fullness of the transmission buffers. In order to maintain a steady data rate, the quality of the image is adjusted better or worse by stepping the quantization parameters. In this way the standard is able to achieve the high compression necessary for video transmission and transmit the best picture possible given the bandwidth of the channel.

A.4.2 MPEG Video Compression Standard

The MPEG video compression standard is a move to standardize the video compression algorithm in order to facilitate growth and interoperability of digital video telecommunications. The algorithm is being proposed by the International Standards Organization (ISO) and is intended for the data rate of 1.5 Mbits/s. A further proposal MPEG II is being proposed for the data rate of 3 - 10 Mbits/s for the compression of CCIR-601 images (MPEG II is rumored to be the compression standard choice for the new U.S. HDTV cable system). The rate of 1.5 Mbits/s is currently special because that is the data rate at which CD-ROMs and DATs can produce data.

The MPEG algorithm uses the JPEG and Px64 standards as a starting point, and many of the same techniques are integrated into the MPEG standard. The MPEG algorithm however, must also meet the special requirement of video and multimedia applications (i.e., a reverse mode and synchronized audio). The basic algorithm is very similar to that of the Px64 standard utilizing the same YUV space, the same image size and, the same frame rate. The motion estimation for the MPEG algorithm however, is performed on 16 by 16 blocks instead of the 8 by 8 blocks used by Px64. The spatial reduction though, is still performed on 8 by 8 blocks. The primary difference in the MPEG algorithm is in the way in which it handles temporal reduction. The MPEG algorithm makes use of three frame types I, P and B, for intrapictures, predicted pictures and bi-directionally interpolated pictures. Intrapictures utilize no temporal reduction and provide access points for random access, but only with moderate compression. Predicted pictures are coded with a reference to past pictures and provide better compression than intrapictures but worse compression than bi-directionally interpolated pictures. Both intrapictures and predicted pictures can be used as a reference for predicting future pictures. Bi-directionally interpolated pictures provide the highest compression rate but, require both a past and a future reference in order to interpolate the picture. Bi-directionally interpolated pictures may not be used as a reference for future pictures. The actual ordering and frequency of these frames is left to the manufacturer but a typical ordering may look like: IBBPBBPBB...IBBPBB. The intrapictures are coded in a manner similar to the JPEG algorithm and the predicted pictures are coded in a manner similar to the Px64 standard. The bi-directionally interpolated pictures however, are interpolated based on a motion vector relative to both the previous and the future reference point. The difference between the prediction and the actual image is then handed to the spatial reduction routines to be encoded in a manner similar to the JPEG algorithm. Other differences exist between MPEG and the Px64 standard but most are just implementation details, rather than fundamental compression differences, and therefore will not be covered here. The audio compression techniques in the MPEG algorithm are also beyond the scope of this document and will not be covered, suffice it to say that they deliver synchronized CD quality sound using only about .25 Mbits/s of the bandwidth.

A.5 Conclusion

Image compression schemes can be seen to come in a wide variety of formats and implementations. Techniques ranging from the original run-length encoding to the latest fractal transforms and video encoding are currently in use. The explosion of digital multimedia in recent years has spurred the field and, is responsible for the development of new and unique mechanisms for compression. This is currently a fast moving field that is and will continue to provide remarkable gains for image transmission and storage.

B Image File Format List and Descriptions

Image file formats, while not directly related to image compression, are one of the largest users of image compression technology. It is then useful to survey the image file formats to get a flavor for what image compression schemes are actually in use in the real world.

Image file formats come in a bewildering array of styles, due to the fact that the majority of imaging applications have been developed "in house" with no special need to interchange the imaging data with other locations. Several standards have been presented to reduce this and, to promote the proliferation of information. But to date, no one standard has emerged. The following is a partial list of some of the more prevalent file formats.

- *CCITT* (Facsimile formats). Described above.
- *CGM* (Computer Graphics Metafile). Developed for the storage and interchange of graphics images. It defines primitives like polygons, etc.
- *DIB* (Microsoft Windows Device Independent Bitmap). Developed by Microsoft for the IBM PC platform. Optionally uses a simple run-length encoding scheme for compression.
- *DXF* (Drawing Interchange Format). Developed by Autodesk for CAD applications. Stores geometric entities rather than actual pixel values.
- *FITS* (Flexible Image Transport System). Used for storing astronomical images. No compression is used.
- *Gem bit image*. For storage and transfer between GEM applications on the IBM PC and Atari platforms. Previously owned by DEC, now is owned by Novell. Compression utilizes run-length encoding of both single values and of patterned runs.

- *GIF* (Graphics Interchange Format). A file format made popular by the CompuServe network. For compression it supports variable length LZW encoding.
- *HDF* (Hierarchical Data Format). Designed at the National Center for Supercomputing Applications to support data sharing among the supercomputing community. For compression it supports run-length encoding and an IMCOMP arial averaging scheme.
- *HP-GL* (Hewlett Packard Graphics Language). Developed by Hewlett Packard for pen plotters and laser printers. Primarily used for line drawing, stored values indicate pen motions rather than actual color values.
- *IFF/ILBM*. Used for multimedia purposes by Electronic Arts. Uses a simple run-length encoding scheme for compression.
- *IPI* (Image Processing and Interchange Standard). An ISO standard.
- *JPEG*. Described above.
- *Kodak Photo CD*. Developed by Kodak for the storage of images in a CD ROM format.
- *Lotus PIC*. Developed by Lotus as an intermediate file between Lotus 1-2-3 and graph printing programs. Stored values indicate graphing primitives like move, draw and fill rather than actual color values.
- *MacPaint*. Native format for Macintosh MacPaint. It uses the PackBits compression scheme described in Section A.2.1.
- *MPEG*. Described above.

- *PBM* (Portable Bitmap Utilities). Variations include the newer PBM+, Portable Gray-scale Map (PGM) and Portable Pixel map (PPM). Shareware format available on UNIX and IBM PCs. No compression is used.

- *PCL* (Hewlett Packard Printer Control Language). Developed by HP for control of HP Laserjet and Deskjet printers. For compression it supports run-length encoding, delta row compression (which only sends changes from one row to the next) and, adaptive compression (which combines all of the above modes along with row skipping and row duplication).

- *PCX.* A proprietary format developed for a PC-based paint program. It uses a simple run-length compression scheme.

- *Postscript.* Developed by Adobe for Macintosh, IBM PCs and UNIX. Postscript is more of a language than a pixel-wise file format and therefore contains no compression.

- *PICT* (QuickDraw Picture Format). Developed for Macintosh QuickDraw pictures. Uses a simple run-length encoding scheme.

- *Sun Rasterfiles.* Used for image storage on Sun systems. For native compression it utilizes a simple run-length encoding scheme but, it also allows for the embedding of TIFF, IFF and other image file formats which may use more advanced compression schemes.

- *TIFF* (Tag Image File Format). An attempt to produce a standard for image file formats. It attempts to be all things to all people and tends to be bogged down by a bulky implementation. For compression, the baseline model supports run-length, Huffman, LZW, CCITT T.4 and T.6.

- *Truevision Targa.* Developed by Truevision for the IBM PC and Macintosh platforms. For compression it utilizes a simple run-length encoding scheme.
- *UNIX plot format.* Used on UNIX for image drawing programs. Stored values indicate line drawing primitives like move, draw and arc rather than actual color values.
- *Utah Raster Toolkit's RLE format.* Used extensively here at the University of Oregon. It utilizes a simple run-length encoding scheme for compression.
- *WMF* (Microsoft Windows Metafile). Developed by Microsoft and contains a display list for Microsoft Windows. No compression.
- *X Window bitmaps.* Used for icons and cursors under X. No compression is used.
- *X Window Dump.* Used to save and restore screen window images under X. No compression is used.

References

Albert, A. E. and Gardner Jr., L. A., *Stochastic Approximation and Nonlinear Regression*, M.I.T. Press: Massachusetts, 1967.

Bodewig, E., *Matrix Calculus*, North Holland Publishing Company: Amsterdam, 1956.

Chellappa R., *Digital Image Processing*, IEEE Computer Society Press: California, 1992.

Cook, R. L., "Stochastic Sampling in Computer Graphics," *ACM Transactions on Graphics*, Vol 5, No 1, pp. 51-72, 1986.

Dippe', M. A. and Wold, E. H., "Antialiasing Through Stocastic Sampling," *Computer Graphics, Annual Conference Series*, ACM SIGGRAPH, pp. 69-78, 1985.

Foley, J., van Dam, A., Feiner, S. and Hughes, J., *Computer Graphics* Addison-Wesley: Massachusetts 1993.

Gondek, J. S., Meyer, G. W., and Newman, J. G., "Wavelength Dependent Reflectance Functions," *Computer Graphics, Annual Conference Series*, ACM SIGGRAPH, pp. 213-220, 1994.

Gray, R. M., "Vector Quantization," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, pp 4-29, 1984

Hamming, R. W., *Digital Filters*, Prentice-Hall: New Jersey, 1977.

Kajiya, J. T., "The Rendering Equation," *Computer Graphics, Annual Conference Series*, ACM SIGGRAPH, pp. 143-150, 1986.

Kay, D. and Levine, John., *Graphics File Formats* Wincrest/McGraw-Hill: Pennsylvania 1992.

Lee, M. E., and Redner, R. A., "A Note on the Use of Nonlinear Filtering in Computer Graphics," *IEEE Computer Graphics and Applications*, Vol 10, No 3, pp. 23-29, 1990.

Lee, M. E., Redner, R. A. and Uselton, S., "Statistically Optimized Sampling for Distributed Ray Tracing," *Computer Graphics, Annual Conference Series*, ACM SIGGRAPH, pp. 61-68, 1985.

LeGall, D., "MPEG: A Video Compression Standard for Multimedia Applications," *Communications of the ACM* Vol 34, No 4, pp. 46-58, 1991.

Liou, M., "Overview of the px64 kbit/s Video Coding Standard," *Communications of the ACM* Vol 34, No 4, 1991.

Mitchell, D. P., "Generating Antialiased Images at Low Sampling Densities," em Computer Graphics, Annual Conference Series, ACM SIGGRAPH, pp. 65-72, 1987.

Oppenheim, A. V. and Willsky, A. S., *Signals and Systems* Prentice-Hall: New Jersey 1983.

Painter, J. and Sloan, K. "Antialiased Ray Tracing by Adaptive Progressive Refinement," *Computer Graphics*, Vol. 23, No. 3, pp. 281-288, 1989.

Peitgen, H., Jurgens, H.and Saupe, D., *Chaos and Fractals - New Frontiers of Science*, Spriger-Verlag, Inc.: New York, 1992

Pennebaker, W. B., Mitchell, J. L. Langdon Jr., G. G. and, Arps, R. B., "An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder," *IBM Journal of Research and Development* Vol. 32, No. 6 pp. 717-726, 1988.

Pennebaker, W. B. and Mitchell, J. L., "Probability Estimation for the Q-Coder," *IBM Journal of Research and Development* Vol. 32, No. 6 pp. 737-752, 1988.

Powell, M. J. D., "A Theorem on Rank One Modifications to a Matrix and its Inverse," *Computer Journal*, Vol. 12, pp. 288-290, 1969.

Pratt, W. K., *Digital Image Processing*, Second Edition, John Wiley and Sons, 1991.

Rao, K. R. and Yip, P., *Discrete Cosine Transform*, Academic Press: Boston, 1990.

Rushmeier, H. E. and Ward, G. J., "Energy Preserving Non-Linear Filters," *Computer Graphics, Annual Conference Series*, ACM SIGGRAPH, pp. 131-138, 1994.

Schreiber, W. F., *Fundamentals of Electronic Imaging Systems*, Springer-Verlag: New York, 1993.

Stark, P. A., *Introduction to Numerical Methods*, Macmillan Publishing Co.: New York, 1970.

Wallace, G. K., "The JPEG Still Picture Compression Standard," *Communications of the ACM* Vol 34, No 4, 1991.

Watt, A. and Watt, M., *Advanced Animation and Rendering Techniques* Addison-Wesley: England 1992.

Welch, T. A., "A Technique for High-Performance Data Compression," *Computer* Vol 17, No 6, pp 8-19, 1984.

Whitted, T., "An Improved Illumination Model for Shaded Display," *Graphics and Image Processing*, Vol 23, No 6 pp. 343-349, 1980.

Witten, I. H., Neal, R. M. and Cleary, J. G., "Arithmetic Coding for Data Compression," *Communications of the ACM*, Vol 30, No 6 pp. 520-540, 1987.

Woods, J. W. and O'Neil, S. D., "Subband Coding of Images," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol ASSP-34, No. 5, 1986.

Ziv, J. and Lempel, A., "Compression of Individual Sequences via Variable-Rate Coding," *IEEE Transactions on Information Theory*, Vol. IT-24, No. 5, pp. 530-536, 1978.

References Available From Other Sources

- The comp.compression forum on the internet provided an invaluable source of compression information especially the Frequently Asked Questions (FAQ) document. The FAQ is posted to the forum or available via anonymous ftp at rtfm.mit.edu: /pub/usenet/news.answers/compression-faq.

- The CCITT standards documents for the T.4 and T.6 specs are available via ftp at src.doc.ic.ac.uk: /computing/ccitt.ccitt-standards/ccitt/1988/ascii in the files 7_3_01.txt.Z and 7_3_02.txt.Z respectively.
- The JBIG algorithm is described in ISO/IEC CD 11544, contained in document ISO/IEC JTC1/SC2/N2285, available from ANSI at (212) 642-4900.
- Description of graphics file formats including TIFF, GIF, FITS, etc., are available via ftp at zamenhof.cs.rice.edu:/pub/graphics.formats or from ftp.ncsa.uiuc.edu: /misc/file.formats/graphics.