Building the BIG Picture: Enhanced Resolution from Coding

by

Roger George Kermode

B.E. (hons) Electrical Engineering,
University of Melbourne, 1989

B.Sc. Computer Science,
University of Melbourne, 1990

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
in Partial Fulfillment of the requirements of the degree of

MASTER OF SCIENCE IN MEDIA ARTS & SCIENCES
at the
Massachusetts Institute of Technology
June 1994

Signature of Author

Program in Media Arts and Sciences
May 6, 1994

Certified by

Andrew B. Lippman
Associate Director, MIT Media Laboratory
Thesis Supervisor

Accepted By

Stephen A. Benton
Chairperson
Departmental Committee on Graduate Students
Program in Media Arts and Sciences

Building the BIG Picture: Enhanced Resolution from Coding

by
Roger George Kermode

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning,
on May 6, 1994
in Partial Fulfillment of the requirements of the degree of

MASTER OF SCIENCE IN MEDIA ARTS & SCIENCES

Abstract

Video Coding Algorithms based on Hybrid Transform techniques are rapidly reaching a limit in compression efficiency. A significant improvement requires some sort of new representation -- a better model of the image based more closely on how we perceive it. This thesis proposes a coder that merges psycho-visual phenomena such as occlusion and long-term memory with the architecture and arithmetic processing used in a high quality hybrid coder (MPEG). The syntactic and processing overhead at the decoder is small when compared to a standard MPEG decoder.

The final encoded representation consists of a small number of regions, their motion, and an error signal that corrects errors introduced when these regions are composited together. These regions can be further processed in the receiver to construct a synthetic background image that has the foreground regions removed and replaced, where possible, with revealed information from other frames. Finally, the receiver can also enhance the apparent resolution of the background via resolution out of time techniques. It is anticipated that this shift of intelligence from the encoder to the receiver will provide a means to browse, filter, and view footage in a much more efficient and intuitive manner than possible with current technology.

Thesis Supervisor: Andrew B. Lippman
Title: Associate Director, MIT Media Laboratory

This work was supported by contracts from the Movies of the Future and Television of Tomorrow consortia, and ARPA/ISTO # DAAD-05-90-C-0333.

Building the BIG Picture: Enhanced Resolution from Coding

by
Roger George Kermode

The following people served as readers for this thesis:

Reader:

Edward H. Adelson
Associate Professor of Vision Science
Program in Media Arts and Sciences

Reader:

Michael J. Biggar
Principal Engineer, Customer Services and Systems
Telecom Australia Research Laboratories, Telstra Corporation Ltd.

To my family;

George, Fairlie, Ivy May, and Meredith,

whose love, support, and understanding made this thesis possible.

.

Contents

List Of Figures

Chapter 1

Introduction

1.1 Motivation

Hybrid transform video compression coders have been the subject of a great deal of research in the past ten years, and within the last three, they have become practical and cost effective for applications ranging from videotelephony to entertainment television. They are the object of optimization rather than basic research. In fact, the sheer size of their expected deployment will effectively freeze their design until general purpose processors can perform fully programmable decoding. The current state of the art provides compression ratios on the order of 50 to 100:1 with acceptable loss of picture quality for home entertainment.

Drawing a parallel with word processing, one could say that the level of sophistication of these coders is currently on a par with word processors of 15 years ago. These early word processors stored text very efficiently using codes of 7 to 8 bits per character, but unfortunately the final representations included little information about the text itself. No information concerning typefaces, page layout, or other higher levels of abstraction was included in the representation. Today the majority of word processors allow one to format text with a veritable plethora of options ranging from the size and style of font to automatic cross referencing between text in different documents. It is easy to see that embellishing the presentation with information about the text has made today's word processors infinitely richer and more useful than those that use words alone.

When one examines encoded video representations today one finds that they are at the 'text only' stage. Current algorithms do a very good job at reducing the size of representation, but in doing so lose almost all ability to manipulate the content in a meaningful fashion. The much heralded information super highway with multi-gigabit data paths is expected to become a reality soon. When it does, it will be crucial that new representations be found for video that enable one to make quick decisions about what to watch without having to search through every single bit.

These representations should contain information about the content that enable machine and viewer alike to make quick decisions without having to examine every second of video. Without these representations the benefits of having access to an information super highway, as opposed to access to traditional media sources will be minimal for all but the expert user. This being the case, it will be difficult to attract people to use the highways and it may become difficult to justify their cost.

Admittedly these are very lofty goals, the amount of effort required to achieve them is enormous. Given the amount of effort required, and the newness of the super highway as a medium, it is likely that the realization of these goals will take quite some time. Consequently, if for no other reason than cost, it is also likely that the initial steps towards making these representations a reality will be derivatives of current representations.

1.2 The problem

Before a new representation can be derived it is useful to examine current state of the art video coding algorithms in order to determine what their good and bad points are. The salient features of current hybrid transform algorithms can be summarized as follows;

- Heavy reliance on statistical models of pictures, in particular spatiotemporal proximity between frames.
- The model used for predicting subsequent frames consists of only one or two single frames that have previously been encoded.
- Frames are subdivided into blocks which are then sequentially encoded using a single motion vector for each block and an error residual.
- Image formation by 3D to 2D projection is ignored; little consideration is given to phenomena such as occlusion, reflection, or perspective.
- There is no easy means for determining the contents of a frame without decoding it in its entirety.

Current research into new approaches to coding generally attempt to address different sets of applications, or use radically different image models. For example, the MPEG[1] group, which has issued one standard for multimedia and broadcast television (MPEG-1) and is putting the final touches on another (MPEG-2 [3]), is beginning a three-year effort deliberately designed to speed the evolution of radically different coders that break the hybrid/DCT mold. These are expected to be used primarily in extremely low-bandwidth applications, but the development spurred by the MPEG effort may extend itself past that.

1. Moving Picture Experts Group, ISO/IEC JTC1 SC29/WG11

As may be surmised from the list of hybrid transform coder features on the previous page, these new modelling techniques derive from either new work on understanding human vision (and mapping that into a compression system), or work on new computational efficiencies that permit complex models to be exploited that previously had been academic tours de force.

1.3 Approach

This thesis proposes an object-oriented coding model that is an architectural extension to existing MPEG-style coders. The objective model assumes no knowledge other than data contained in the frames themselves (no hand-seeded model of the scene is used), and is basically two or two-and-a-half dimensional (two dimensional plus depth of each two dimensional layer). The model classifies objects within the frame on the basis of their appearance, connectivity, and motion relative to their surrounds. While the raster-oriented decomposition of the frame (as used in MPEG) remains in this coder, the image is no longer coded as blocks classified on an individual basis. Instead, the image is coded as a montage of semi-homogenous regions in conjunction with an error signal for where the model fails.

The structure of the coder borrows many elements from previous coders and image processing techniques in particular MPEG 2 [3], Teodosio's Salient Still [28] and the work by Wang, Adelson and Desai [4], [30]. The resulting coding model incorporates the efficiencies of motion compensation using 6 parameter affine models as well as the increased accuracy of standard block based motion compensation for foreground regions that do not match the affine model. It is hoped that the coding scheme developed here may eventually be extended into the framework of a fully three dimensional representation as developed

by Holtzman [13] which promises extremely high compression ratios. The genealogy of the Enhanced Resolution Coder is shown in Figure 1.1.

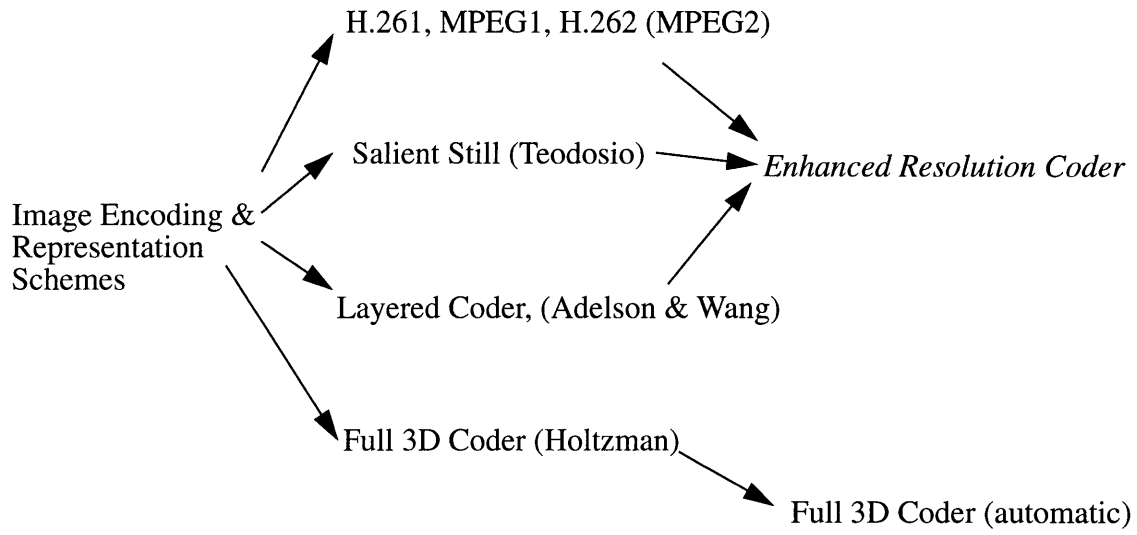


Figure 1.1. Genus of the Enhanced Resolution Codec

Another way to classify the algorithm is to consider its position in the table below,

Encoder Type	Information Units	Example
Waveform	Color	D1
Transform	Blocks	JPEG [1]
Hybrid Transform	Motion+ Blocks	MPEG [3], [12]
2D Region Based	Regions + Motion + Blocks	Mussman's Work [21] Layered Coder [4], [30] Enhanced Resolution Coder
Scenic	3D Objects	"Holtzman Encoder" [13]
Semantic	Dialogue, Expressions	Screenplay

TABLE 1.1. Encoder Classes (after Mussman)

Another motivating advantage of such a structural analysis of the image is that the structure can be exploited at the receiver, without any additional recognition, to facilitate browsing, sorting sequences, and selecting items of interest to the viewer. In essence, the pictures are transmitted as objects that are composited to form a frame on decoding, and

each picture carries information about the scene background, the remaining foreground objects, and the relation between them. As compressed video becomes pervasive for entertainment, it is anticipated that "browsability" of the sequence will become at least as important as the compression efficiency to the viewer, for there is little value in having huge libraries of content without a means to quickly search through them to find entries of interest.

With these ideas in mind, the target application of the televised lecture was chosen to demonstrate the enhanced resolution codec. The reasons behind this decision are threefold. First, the scene is best photographed with a long lens that minimizes camera distortion (this is a requirement for optimal performance). Secondly, enhancing the resolution of written or projected materials in the image will allow the viewer to see the material at a finer resolution, while still seeing a broader view of the lecture theatre. In fact the algorithm will allow the viewer to determine whether the entire scene or just the written/projected material is displayed. This feature is particularly useful as it allows the user to choose the picture content according to interest and decoder ability (small vs large screen). Finally, apart from technical considerations the aforementioned application has the potential to provide a useful service to students who, for whatever reason, have insufficient teaching facilities available, thus allowing them to participate in lectures from other schools and institutions[1].

The remainder of this thesis is organized as follows. Chapter Two briefly reviews the fundamentals of encoding still and moving images. Chapter Three examines three recent advances in image representation. An analysis of the theory behind resolution enhance-

1. One such program of televised lectures is already been undertaken by Stanford University which regularly broadcasts lectures via cable to surrounding institutions and companies who have employees enrolled in Stanford' classes. Livenet in London, England and Uninet in Sydney, Australia are also other examples of this type of program.

ment is presented in Chapter Four. Chapters Five and Six map the results of the previous chapters into a frame work that defines the new encoder. Simulation results of the new algorithm, including a comparison with the MPEG1 algorithm, are presented in Chapter Seven. Finally, Chapter Eight presents an analysis on the performance of the new algorithm and makes suggestions for future work.

Necessary and self contained technical derivations of various mathematical formulae and modifications to existing algorithms have been moved to the Appendices.

Chapter 2

Image Coding Fundamentals

Currently all displayed pictures, whether static or moving, are comprised of still (stationary) images. For example, photographs, television images and movies are sequences of one or more still images. This representation has not come about by accident but instead by design. As one Media Lab professor is fond of saying "God did not invent scanlines and pixels....people did". Therefore, it is not surprising that the current image coding techniques for moving tend to be based on philosophies very similar to those used for encoding still images, mainly that of spatio temporal redundancy and probabilistic statistical modeling. This chapter examines these philosophies as they have been applied in development of the JPEG[1] and MPEG standards for still and moving image coding respectively.

2.1 Representation

The choice of representation affects two fundamental characteristics of the coded image;

- Picture Quality
- Redundancy present in the representation

These two characteristics are independent up to a point, generally one can remove redundant information from a representation and not suffer any loss in picture quality, this is *lossless* compression. However, once one starts to remove information beyond that which is redundant the picture quality degrades. This type of compression, where non redundant information is deliberately removed from the representation, is known as *lossy*

1. Joint Photographic Experts Group (JPEG), of ISO/IEC JTC1/SC18/WG8

compression. The amount and nature of the degradation caused by the use of lossy compression techniques is highly dependent on the representation chosen. Hence, it makes sense that great care should be exercised in choosing an appropriate representation when using lossy compression techniques.

2.2 JPEG Standard [1]

The Joint Photographic Experts Group (JPEG) standard for still image compression is fairly simple in its design. It is comprised of three main functional units; a transformer, a quantizer, and an entropy coder. The first and last of these functional units represent lossless processes[1] with the second block, quantization, being responsible for removing perceptually irrelevant information from the final representation. Obviously, if one desires high compression ratios then more information has to be discarded and the final representation becomes noticeably degraded when compared with the original.

2.2.1 Transform Coding

A salient feature of most images is that they exhibit a high spatial correlation between the pixel values. This means that representations in the spatial domain are usually highly redundant and, therefore, require more bits than necessary when encoded. One method which attempts to decorrelate the data to facilitate better compression during quantization is Transform Coding. Transform Coding works by transforming the image from the spatial domain into an equivalent representation in another domain (e.g. the frequency domain). The transform used must provide a unique and equivalent representation from which one can reconstruct all possible original images. Consider the general case where a single one dimensional vector $x = [x ,..., x_k]^T$ is mapped to another $u = [u ,..., u_k]^T$ by the transform T,

1. With the exception of errors due to finite arithmetic precision

$$u = Tx \tag{1.1}$$

The rows of the transform T, t_i are orthonormal, that is they satisfy the constraint

$$t_i t_j^T = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{if } i = j \end{cases}$$

$$\text{or} \quad TT^T = I$$

$$\text{i.e.} \quad T^{-1} = T^T \tag{1.2}$$

The vectors $t_1, \ldots, t_k$ are commonly called the basis functions of the transform T. The fact that they form an orthonormal set is important as it ensures that;

- they span the vector space of x in a unique manner, and
- the total energy in the transform u is equal to that of the original data x.

In other words, this property ensures that there is a unique mapping between a given transform u and the original data x[1].

Transforming from the spatial domain into another domain gains nothing in terms of the number of bits required to represent the information. In fact, the number of bits required to represent the transformed information can increase. The gain comes in that the transformed representation can be quantized more efficiently. This is due to the fact that the transform coefficients have been decorrelated and have also undergone energy compaction, thus many of them can be quantized very coarsely to zero.

1. The mathematics above deal with data that is represented by one dimensional vectors, however pictures are two dimensional entities. This is not a problem as one can simply extend the concept of transformation into two dimensions by applying a pair of one dimensional transforms performed horizontally and vertically (or vice versa).

JPEG uses the Discrete Cosine Transform (DCT) to reduce the amount of redundancy present in the spatial domain. The image to be coded is subdivided into small blocks(8 by 8 pixel) which are then individually transformed into the frequency domain. Unlike the Discrete Fourier Transform (DFT) which generates both real and imaginary terms for each transformed coefficient, the DCT generates real terms when supplied with real terms only. Thus, the transform of an 8 by 8 pel block yields an 8 by 8 coefficient block, there is no expansion in the number of terms. However, these terms require more bits, typically 12 bits to prevent round off error as opposed to 8 in the spatial domain, so the representation is in fact slightly larger.

The formulae for the N by N two dimensional DCT and Inverse DCT (IDCT) are

DCT

$$F(u,v) = \frac{2}{N} C(u)\, C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos\left(\frac{(2x+1)\, u\pi}{N}\right) \cos\left(\frac{(2y+1)\, v\pi}{N}\right)$$

IDCT

$$f(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u)\, C(v)\, F(u,v) \cos\left(\frac{(2x+1)\, u\pi}{N}\right) \cos\left(\frac{(2y+1)\, v\pi}{N}\right)$$

with $u, v, x, y = 0, 1, 2, \ldots N-1$

where x, y are coordinates in the pixel domain
u, v are coordinates in the transform domain

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for u, v = 0} \\ 1 & \text{otherwise} \end{cases} \qquad \textbf{(1.3)}$$

The means by which a gain in compression efficiency can be achieved (by subsequent quantization) lies in the distribution of the transformed coefficients. A corollary of the fact that real world images exhibit high spatial correlation, is that the spectral content of real world images is skewed such that most of the energy is contained in the lower frequencies. Thus, the DCT representation will tend to be sparsely populated and have most of the energy in the coefficients corresponding to lower frequencies. This observation becomes

intuitively obvious when one examines the basis functions for the DCT as shown in Figure 2.1.

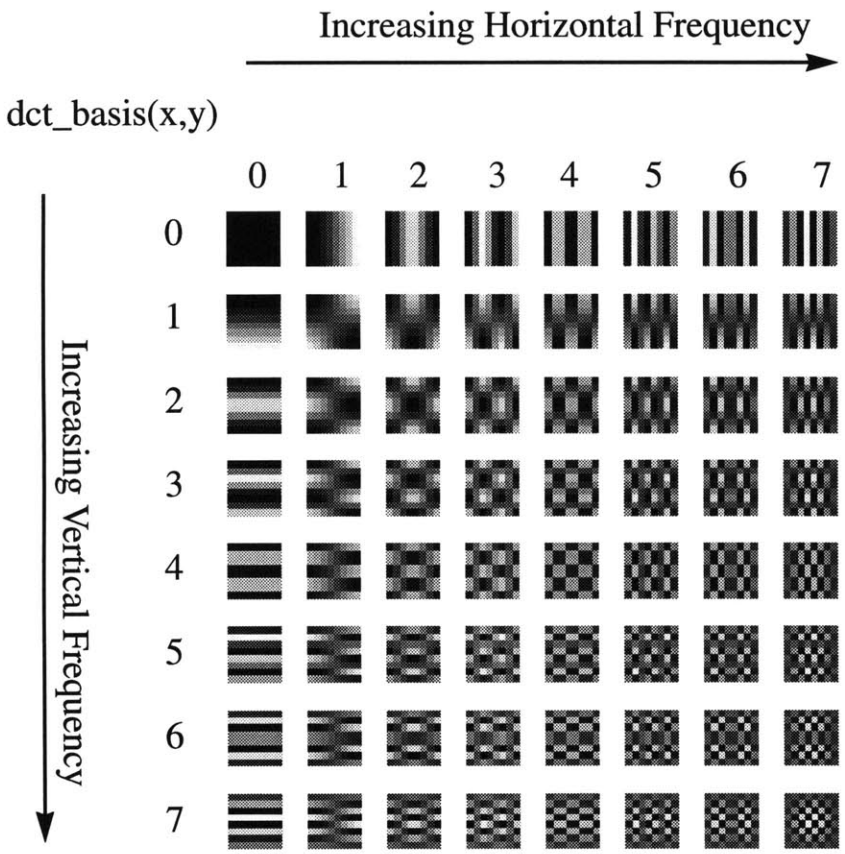


Figure 2.1 Basis Functions for an 8 by 8 Two Dimensional DCT

The basis functions depicted towards the upper left corner correspond to lower frequencies while those towards the lower right correspond to higher frequencies. The magnitude of the transform coefficient corresponding to each of the basis functions describes the 'weight' or presence of each basis function in the original image block.

2.2.2 Quantization of Transform Coefficients

Having transformed the image into the frequency domain JPEG next performs quantization on the coefficients of each transformed block.

The coefficients corresponding to the lower frequency samples play a larger role in reconstructing the image than the coefficients corresponding to higher frequencies. This is true not only by virtue of the fact that they are much more likely to be present as stated in Eq. 2.2.1, but also because perceptually the human visual system is much more attuned to noticing the average overall change in image intensity between blocks as opposed to the localized high frequency content contained in the higher frequency coefficients[1]. Thus, a given distortion introduced into a component that represents low frequencies will be much more noticeable than if the same distortion was introduced into a component that represents high spatial frequencies.

The DC (zero frequency) coefficient is particularly susceptible to distortion errors, as an error in this coefficient affects the overall brightness of the block. For this reason the DC coefficient is quantized with a much finer quantizer than that used for the remaining AC (non zero frequency) coefficients. Likewise, lower frequency AC components are quantized less aggressively than higher frequency ones.

2.2.3 Entropy Coding

The final stage of the JPEG compression algorithm consists of an entropy coder. Entropy is generally defined to be a measure of (dis)order or the amount of redundancy present in a collection of data. A set of data is said to have high entropy if its members are independent

1. (Having said this it must be noted that if an image composed of inverse transformed blocks is displayed where the high frequency coefficients have been removed from all but a few of these blocks then those few blocks will be highly noticeable. For this reason it is desirable to attempt to discard the same number of higher coefficients in each block.)

or uncorrelated. Conversely, a set that displays high dependency or correlation between its members is said to have low entropy. Given a set of symbols and their relative probabilities of occurrence within a message, one can make use of this concept in order to remove redundancy in their encoded representation.

First, consider a set of data which is completely random, in other words one where every symbol is as likely as another to be present in a coded bitstream. Suppose that a restriction is imposed which states that each symbols' encoded representation is the same length. As each symbol is the same length in bits and just as likely to occur as another they all cost the same to transmit, the representation is efficient. Now consider the case where the symbol probabilities are not equal. It is easy to see that even though certain symbols hardly ever occur they will still require just as many bits as the more frequent ones, therefore, the representation is said to be inefficient.

Now suppose that the restriction that all symbols be represented by codes of the same length is relaxed, thereby allowing codes of different lengths to be assigned to different symbols. If the shorter codes are assigned to more frequently used symbols and the longer codes to the less frequent, then it should be possible to reduce the number of bits required to encode a message consisting of these symbols. However several problems arise out of the fact that the codes are of variable length;

- the position of a code in the encoded bit stream may no longer be known in advance,
- a long code may by missed if it is mistaken for a short code.

As a result, great care must be taken to ensure that no ambiguities arise when assigning the codes. If this requirement is met then the parsing of the bitstream becomes as simple as

knowing where the previous code finished as this automatically provides the starting position for its successor.

JPEG uses the results from a Huffman coding scheme based on this concept. It is useful to introduce two equations ([15],[19],[23]); first the average code word length $\bar{L}$ for the discrete random variable x describing a finite set of symbols is defined as

$$\bar{L} = \sum_{x} L(x)\, p_x(x) \qquad \text{bits/symbol} \tag{1.4}$$

and secondly, the entropy of x, $H(x)$, defined by Eq. (1.5), is the lower bound on $\bar{L}$

$$H(x) = -\sum_{x} p_x(x) \log_2(x) \qquad \text{bits/symbol} \tag{1.5}$$

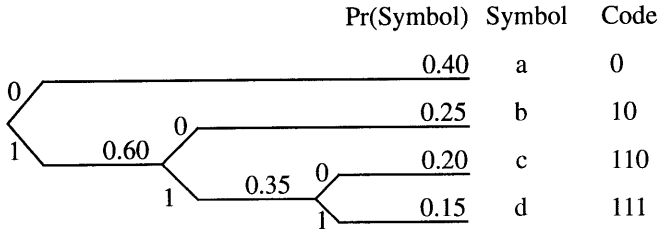
In short, the aim of entropy coding is try to make $\bar{L}$ as close as possible to $H(x)$. If $\bar{L} = H(x)$ then all the redundancy present in x will have been removed and hence any attempt to achieve even greater compression will result in loss of information. This is a theoretical limit and is seldom reached in practice. However it should be noted that higher compression ratios may be achieved without loss by choosing a different representation for the same information. The theoretical limit has not been violated in these cases. What happens is that the pdf of x changes, resulting in a change of $H(x)$ which falls below $H(x)$ of the original representation. Thus it makes sense to choose the most efficient representation before quantizing and subsequent entropy coding.

Huffman coding makes several big assumptions; first, that the probability distribution of the set of symbols to be encoded is known and secondly that the probabilities do not change. Given that these assumptions hold, the codebook containing the code for each symbol can be precomputed and can either be stored permanently in both the encoder and

decoder or transmitted once by the encoder to the decoder at the beginning of a transmission.

The algorithm used to design these codebooks is very simple and constructs a binary tree that has the symbols for leaves and whose forks describe the code. The probability of each symbol is generated by performing preliminary encoding runs and recording the frequency of each symbol in the final representation. Provided sufficiently representative data is used to generate these symbols, the actual probability of occurrence of each symbol may be approximated by its relative probability in the preliminary representation.

For a set of n symbols the algorithm constructs the tree in $n-1$ steps where each step involves creating a new fork that combines the two nodes of highest probability. The new parent node created by the fork is then assigned a probability equal to the sum of the probability of its two children. A simple example is shown below in Figure 2.2



$$\bar{L} = 1x0.40 + 2x0.25 + 3x0.20 + 3x0.15$$
$$= 1.95 \quad \text{bits/symbol}$$

$$H = -(0.40\log_2(0.40) + 0.25\log_2(0.25) + 0.20\log_2(0.20) + 0.15\log_2(0.15))$$
$$= 1.904 \quad \text{bits/symbol}$$

Figure 2.2 Huffman Coding Example

Several codebooks generated by a method similar to that described above are used within JPEG for various types of data in a variety of different situations.

2.3 Frame Rate

When one moves from still images to moving images comprised of a series of still images, one of the main things that must be taken into account is the rate at which pictures are presented to the viewer. If too few pictures are presented to the viewer then any motion within the picture will appear jerky, the viewer will notice that the moving image is not continuous but in fact comprised of many separate still images. However, if the frame rate is increased to a sufficiently high rate then the still images will "fuse" causing motion within the picture to appear smooth and continuous.

The frame rate at which fusion occurs is known as the *critical flicker frequency* (cff) and is dependent upon the picture size, ambient lighting conditions, and the amount of motion within the frame [17] & [23]. For example when a T.V. screen is viewed in a poorly lit room the cfff decreases, conversely the cfff increases when the same T.V. screen is viewed in well lit conditions. The cfff also increases with picture size, which explains why motion pictures are shot at 24 frames per second and then projected at 72 frames per second with each frame being shown three times.

Standard television uses a special method to increase the temporal resolution without increasing the bandwidth of the transmitted signal or decreasing the spatial resolution by an inordinate amount. This trick is known as *interlace* and involves transmitting the picture as two separate fields; the even lines of the picture are sent in the first field, followed by the odd lines in the second field. Transmitting a picture in this manner means that the perceivable spatial resolution of the displayed image is larger than one field alone but smaller than if both fields had been transmitted simultaneously (progressive scanning). The gain comes in that the temporal resolution is increased beyond that of a system where the fields are combined into a single picture. Thus a small sacrifice in spatial resolution is

made to allow for better temporal resolution without increasing the number of scanlines per unit time.

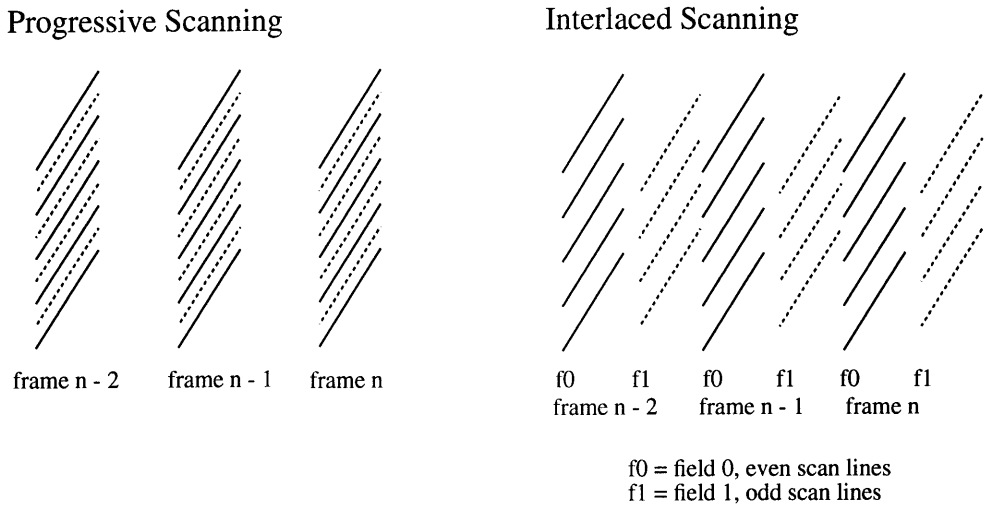


Figure 2.3 Progressive and Interlace Scanning

While interlace is appropriate and useful for analog televisions that only display pictorial information, it is particularly bad for computer generated text or graphics. When a normal television displays these sorts of artificial images the human eye fails to merge the spatial information in successive fields smoothly, and the 30 (NTSC)/ 25(PAL) Hz flicker becomes painfully obvious. For this reason, there has been a concerted push towards higher frame rates both for large screen computer monitors and emerging HDTV system proposals. These HDTV systems will need these higher frame rates to support a larger pictures. In addition, higher frame rate will also be needed to support the anticipated computer driven applications made possible by the digital technology used to deliver and decompress the images they will display. Some common frame rates used within the movie, television and computer industries are 24, 25, 29.97, 30, 59.94, 60 and 72 frames per second.

2.4 MPEG Standards [2], [3]

The Moving Picture Experts Group (MPEG) standard MPEG1 [2] and draft standard MPEG2 [3] are based on the same ideas that were used in the creation of the JPEG algorithm for still images. In fact, MPEG coders contain the same transform, quantization and entropy coding functional blocks as the JPEG coder. The difference between the MPEG and JPEG coders lies in the fact that the MPEG coders exploit interframe redundancy in addition to spatial redundancy, through the use of motion compensation. The three standards can be differentiated as follows;

- JPEG compresses single images independently of one another.
- MPEG1 is optimized to encode picture at CD-ROM rates of around 1.1 Mbit/s at resolution roughly a quarter of that for broadcast TV by predicting the current picture from previously received ones.
- MPEG2 inherits the features present in MPEG1 but also includes additional features to support interlace and was optimized for bit rates of 4 and 9 Mbit/s

2.4.1 Motion Estimation and Compensation

Many methods for estimating the motion between the contents of two pictures have been developed over the years. As motion estimation is only performed in the encoder and hence the method chosen does not impact on interoperability, MPEG does not specify any particular algorithm for generating the motion estimates. However, in an effort to limit decoder complexity and increase representation efficiency, the decision was made that motion vectors should be taken to represent constant (i.e. purely translational) motion for small tiled regions of the screen. The boundaries of these tiles coincided with those for a

grouping of a small number of blocks used by the DCT stage, thus allowing the decoder to decode each tile in turn with a minimal amount of memory. Consequently, the most common method used to generate motion information for MPEG implementations is the block based motion estimator.

Block based motion estimators are characterized by the fact that they generate a single motion vector to represent the motion of small (usually square) group of pels. The motion vector for each block is calculated by minimizing a simple error term describing the goodness of fit of the prediction by varying the translation between the block's position in the current frame and its position in a predictor. The translation corresponding to the minimum error is chosen as the representative motion vector for the block. An example of a prediction made by a typical block based motion estimator is shown in Figure 2.4 below.

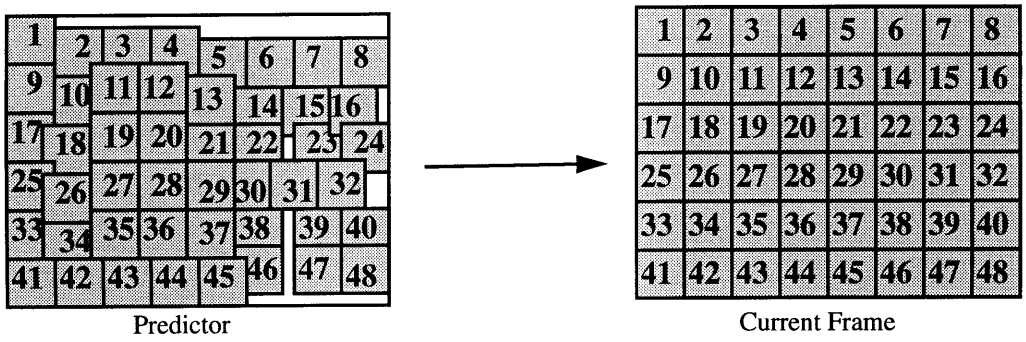


Figure 2.4 Block Based Motion Compensation Example

A common example of an error term used by block based motion estimators is the Sum of Absolute Differences (SAD) between the estimate and the block being encoded. Generally, only the luminance component is used, as it contains higher spatial frequencies than the chrominance components and can, therefore, provide a more accurate prediction of the motion. The SAD minimization formula for a square block of side length *blocksize* is given by

$$SAD(\delta_x,\delta_y) = \sum_{y=0}^{blocksize-1} \sum_{x=0}^{blocksize-1} \text{abs}\,(lum_{current}(x,y) - lum_{pred}(x+\delta_x, y+\delta_y)) \quad (2.1)$$

Where (δ_x,δ_y) is the motion vector which minimizes the Sum of Absolute Differences; $SAD(\delta_x,\delta_y)$

Several advantages and disadvantages result from using the SAD error measure for motion estimation. The main advantage is that the resulting vectors do in fact minimize the entropy of the transform encoded error signal given the constraint that the motion must be constant within each block. However, the hill climbing algorithm implicit with the minimization process means that the motion vectors generated will not necessarily represent the true translational motion for the block, despite the fact that the entropy of the error signal is minimized. This is particularly true for blocks that contain a flat luminance surface or more than one moving object. Given that one is attempting to find true motion, the following list summarizes the main failings of block based motion estimation;

- The motion estimate will most likely be quite different from the true motion where the block's contents are fairly constant and undergo slight variations in illumination from frame to frame. The cause of these errors lies in the fact that small amounts of noise will predominate in the SAD equation causing it to correlate on the noise instead of the actual data.
- Blockwise motion estimation assumes purely translational motion in the plane of the image. It does not model zooms or rotations accurately as these involve a gradually changing motion field with differing motion for each pel within a block

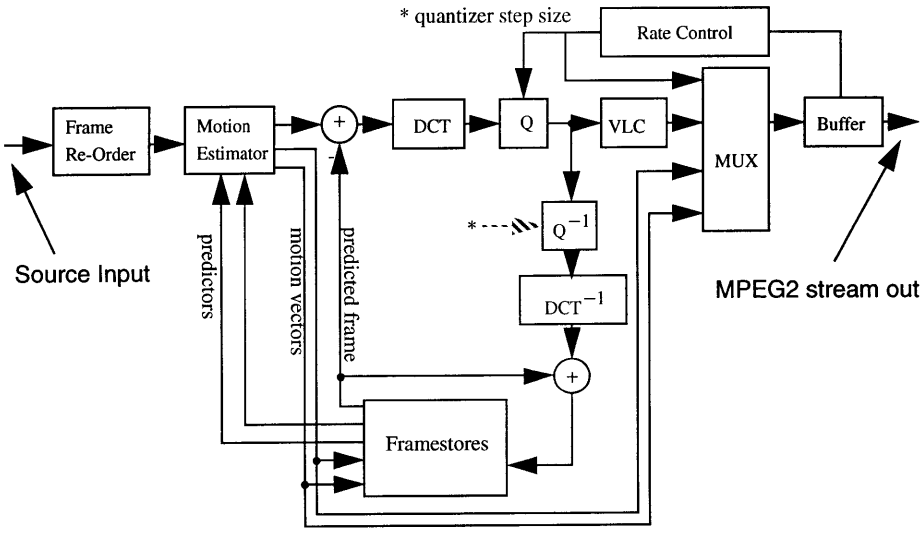
- The motion estimate is deemed constant for the entire macroblock, causing an inaccurate representation when there is more than one motion present in the block or complex illumination changes occur.

Therefore block based motion estimation, while easy to implement, does not necessarily result in either a particularly accurate representation of the true motion, nor a low MSE where non translational motion is present.

2.4.2 Residual Error Signal

Generally, there is some error associated with the prediction resulting from motion estimation and compensation techniques. Therefore, there must be a way to transmit additional information to correct the predicted picture to the original. This error can be coded using the same transform and entropy encoding methods described previously in 2.2.1 to 2.2.3. It should also be noted that there will be places in the picture for which the prediction will be particularly bad, in which case it may be less expensive to transmit that portion without making the prediction, i.e. intra coding.

Figure 2.5 below, shows the main functional blocks within an standard MPEG2 encoder; motion estimator, several frame-stores, 8x8 DCT, quantizer (Q), entropy or VLC encoder, inverse-quantizer (Q^{-1}), 8x8 IDCT, rate-control, and output-buffer.



DCT - 8x8 Discrete Cosine Transform
DCT^{-1} - 8x8 Inverse Discrete Cosine Transform
Q - Quantization
Q^{-1} - Inverse Quantization
VLC - Variable Length Coding

Figure 2.5 MPEG2 Encoder Block Diagram

The corresponding MPEG2 decoder is quite similar. It consists of a subset of the encoder, with the addition of a VLC decoder. It should be noted that the motion estimator, one of the most expensive functional blocks in the encoder, is not required by the decoder. Therefore, the computational power required to implement an MPEG decoder is substantially less than that required for an MPEG encoder. For this reason MPEG is characterized as being an asymmetric algorithm where the requirements of the encoder are substantially different from the decoder. A block diagram of an MPEG decoder is shown in Figure 2.6.

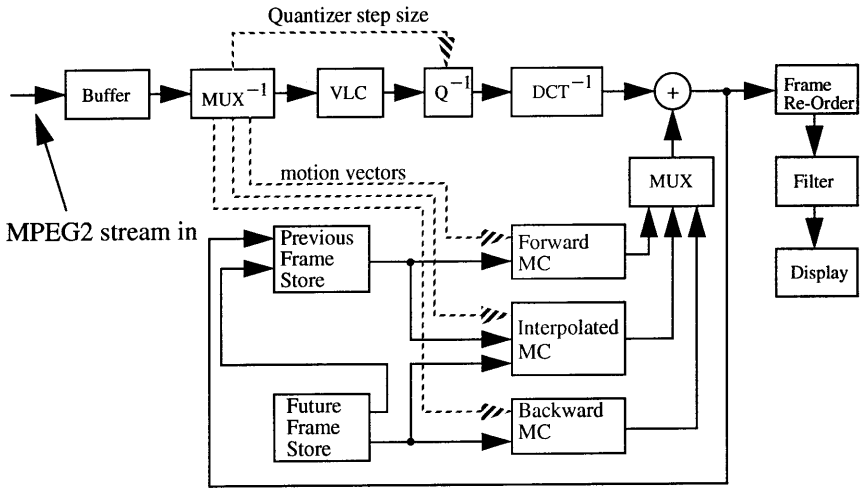


Figure 2.6 MPEG2 Decoder Block Diagram

Chapter 3

Recent Advances in Image Representation

As computing power has increased, the notion of what a digital image representation should encompass has grown to be something more than just a compressed version of the original. This chapter describes three recent approaches to image representation undertaken at the Media Laboratory that have helped to inspire the work in this thesis.

3.1 Salient Stills

The Salient Still developed by Teodosio [28] is a good example of what can be done given sufficient memory and processing time. The basic premise behind the Salient Still is that one can construct a single image comprised of information from a sequence of images, and that this single image contains the salient features of the entire sequence. The Salient Still then has the ability to present not only contextual information in a broad sense (e.g. the audience at a concert) but also the ability to provide extra resolution at specific regions of interest or salient features (e.g. close up information of the performer at the concert). A block diagram showing how a Salient Still is created is shown below.

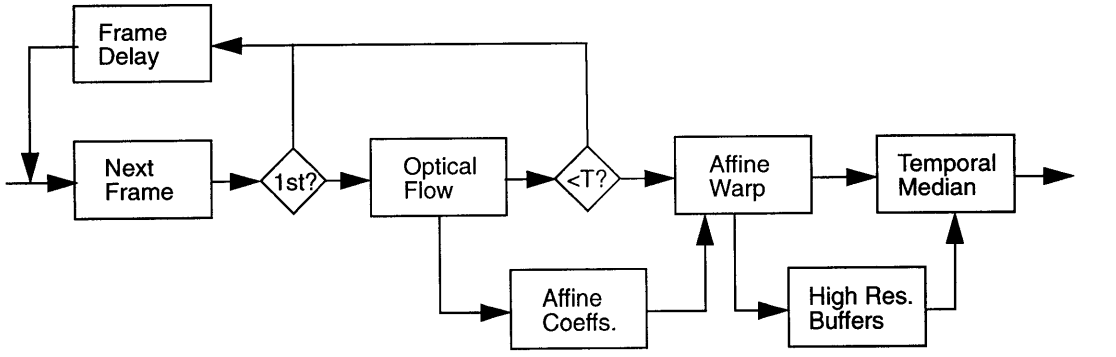


Figure 3.1. Salient Still Block Diagram [28]

The main algorithmic ideas to be gleaned from the Salient Still algorithm are;

- A single image is created by warping many frames into a common space.
- The value of a pel at a particular point is the result of a temporally weighted median of all the pels warped to that position.
- The motion model used to generate the warp is a 6 parameter affine model derived by a Least Squares fit to the optical flow [6].

$$d_x = a_x + b_x x + c_x y$$
$$d_y = a_y + b_y x + c_y y \quad \text{(3.1)}$$

Some of the problems associated with salient generation are;

- Optical flow is particularly noisy near motion boundaries.
- Inaccurate optical flow fields result in an inaccurate affine model.
- The affine model is incapable of accurately representing motion other than that which is parallel to the image plane.

3.2 Structured Video Coding

The "Lucy Encoder" described in Patrick McLean's Master's thesis "Structured Video Coding" [18] explores what is possible if one already has an accurate representation of the background. Footage from the 1950's sitcom "I Love Lucy" provided an easy means to explore this idea for two reasons; first, the sets remained the same from episode to episode and secondly, the camera motions used at the time were limited for the most part to pans. Thus, construction of the background was a fairly easy process. All one had to do was composite several frames together using pieces from each frame where the actors were not

present. In addition to removing the actors, it was also possible to generate backgrounds that were wider than the transmitted image, thus enabling the viewer to see the entire set as opposed to the portion currently within the camera's field of view.

Having obtained an accurate background representation the next step was to extract the actors. This proved to be no easy task. However, with some careful pruning and region growing the actors were successfully extracted (along with a small amount of background that surrounded them). Replaying the encoded footage then became a simple matter of transmitting an offset into the background predictor and pasting the actors into the window defined by this offset. McLean showed that if one transmitted the backgrounds prior to the remaining foreground then it was possible to encode "I Love Lucy" with reasonable quality at rates as low as 64kbits/sec using Motion JPEG (M-JPEG[1]).

Several problems existed in the representation. Zooms could not be modeled as only the motion due to panning was estimated. In addition, the segmentation sometimes did not work as well as it should have, as either not all of an actor was extracted, or it was possible to see where the actors were pasted back in by virtue of mismatch in the background. However, these shortcoming are somewhat irrelevant as McLean demonstrated the more important idea that the algorithm should generate a representation that was not based solely on statistical models but one based on content. The representation now contained identifiable regions that could be attributed to a distinct physical meaning such as a "Lucy" region or a "Ricki" region.

1. Motion JPEG is a simple extension of JPEG for motion video in which each frame consists of a single JPEG compressed image

3.3 Layered Coder

More recently, work in the area of layered coding has been done by Wang, Adelson and Desai [4], [30]. Along the same lines as the Salient Still and the Structured Video Coder, the Layered Coder addresses the idea of modeling not just the background by creating a small number of regions whose motion is defined by a 6 parameter affine model. These regions are transmitted once at the beginning of the sequence with subsequent frames generated by compositing the regions together after they have been warped using the 6 parameter models.

The regions are constructed over the entire sequence by fitting planes to the optical flow field calculated between successive pairs of frames. Similar regions from each segmentation are then merged to form a single region which is representative of its content for the entire sequence.

As might be surmised this technique works best for scenes where rigid body non 3D motion takes place. Another problem lies in the fact that the final regions have an error associated with their boundaries due to the inaccuracies of the segmentation. Thus, when the sequence is composited back together there tends to be an inaccurate reconstruction near the boundaries. Finally, as only a single image is transmitted to represent each region it turns out that closure (i.e. a value for every pel) cannot be guaranteed during the compositing stage after each region has been warped.

For all these reasons the layered coder does not generate pixel accurate replicas of the frames in the original sequence for all but the simplest of scenes. However more importantly, even though a high Mean Square Error (MSE) may exist, the reconstructed images look similar to the original. The error is now, not so much an error due to noise or quantization but, a geometric error associated with an inaccurate motion model.

3.4 General Observations

The three representations described in this chapter share several things in common;

- Many frames are used to construct a single image representative of a region in the scene.
- They construct this image using optical flow [6] to estimate interframe motion.
- One or more 6 parameter affine models are used to represent the interframe motion.

The techniques described above work well if one is attempting to model a simple scene and can generate reasonable approximations to actual regions in a frame. However if pixel accurate replication is required then, in general, the resulting images will not be accurate due to geometric distortions introduced by the affine model's inability to represent 3D motion. Therefore given that an efficient representation is paramount, if one requires a geometrically accurate reconstruction or is modelling a sequence containing significant amounts of 3D motion, then one should use a more accurate motion model in addition to also including an error signal with the affine model to correct these distortions.

One final problem these methods face is that they rely on the accuracy of the motion estimates generated by the optical flow algorithm. Optical flow attempts to determine the velocity of individual pels in the image based on the assumption that the brightness constraint equation below holds.

$$\frac{d}{dt}lum(x,y) = 0 \tag{3.2}$$

This equation can be solved using a number of methods, the two most popular are correspondence methods (which are essentially the same as block matching) and gradient methods which attempt to solve the first order Taylor Series expansion of (3.2),

$$v_x \frac{\partial}{\partial x} lum\,(x, y) + v_y \frac{\partial}{\partial y} lum\,(x, y) + \frac{\partial}{\partial t} lum\,(x, y) = 0 \qquad \textbf{(3.3)}$$

over a localized region R [6]. In order for gradient based optical flow to work the following assumptions are made;

- The overall illumination of a scene is constant,
- The luminance surface is smooth,
- The amount of motion present is small.

When these assumptions are invalid optical flow fails to give the correct velocity estimates. This is particularly true for;

- Large motions,
- Areas near motion boundaries,
- Changes in lighting conditions.

Thus for particularly complicated scenes involving lots of motion or lighting changes the output from the optical flow algorithm will be inaccurate. Hence any segmentation based upon the output from optical flow under these circumstances will be inaccurate.

Chapter 4

Enhancing the Background Resolution

A theme common to the three representations described in the previous chapter was that they all used more than one frame to construct a single representative image. The ability to combine frames of video in this way can result in two benefits. First one can remove foreground objects from the final image. Secondly, under certain conditions it may be possible that the apparent resolution of the final image can be enhanced beyond that of the original images.

This chapter concentrates on the second benefit, specifically, methods that increase the sample density and resolution above that of the transmitted picture through the combination of several pictures sampled at different resolutions and times. The first benefit, that of removing foreground objects from the final image, is the subject of the next chapter.

4.1 One Dimensional Sub Nyquist Sampling

Before examining resolution for images, it useful to examine a simpler one dimensional problem that can later be extended into two dimensions for images. Consider the one dimensional sequence $x[n]$ shown below,

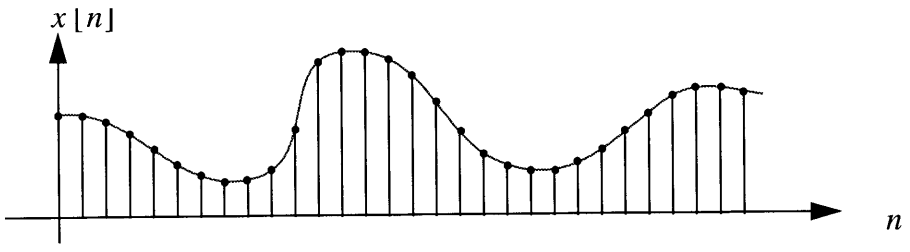


Figure 4.1 An arbitrary one dimensional sequence, $x[n]$

This sequence $x[n]$ can be broken up into two sub nyquist sampled sequences; the first consisting of even samples $x_e[n]$ and the second $x_o[n]$ of the remaining odd samples.

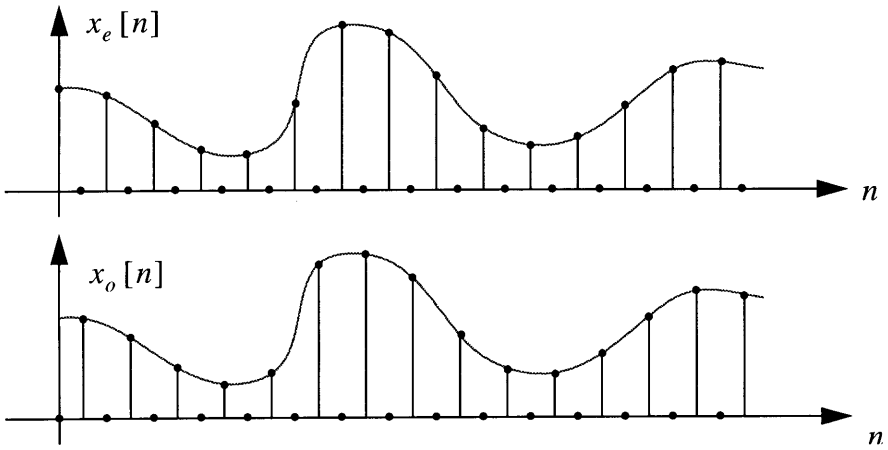


Figure 4.2 Sub nyquist sampled sequences, from odd and even Samples of Figure 4.1

The operations required to remove the even and odd samples from $x[n]$ to generate these sequences are described by the following formulae

$$\begin{aligned} x_e[n] &= \frac{1}{2}\left(x[n] + (-1)^n x[n]\right) \\ x_o[n] &= \frac{1}{2}\left(x[n] - (-1)^n x[n]\right) \end{aligned} \tag{4.1}$$

In the frequency domain these formulae correspond to the introduction of aliased versions of the original sequence's spectrum,

$$\begin{aligned} X_e(\omega) &= \frac{1}{2}\left(X(\omega) + X(\pi-\omega)\right) \\ X_o(\omega) &= \frac{1}{2}\left(X(\omega) - X(\pi-\omega)\right) \end{aligned} \tag{4.2}$$

Furthermore, it easy to see that when these two sequences are added together the aliased components cancel each other and the original sequence is obtained.

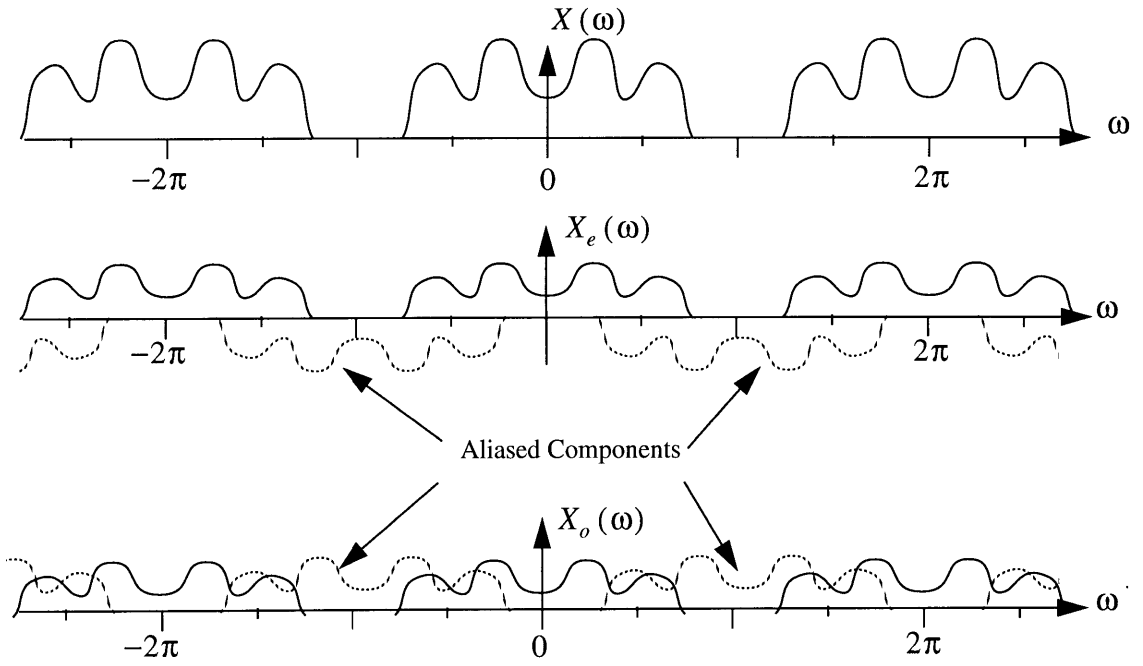


Figure 4.3 Spectral Analysis of Down Sampling

4.2 Enhancing Resolution Using Many Cameras

So far it has been shown that one can decompose a one dimensional sequence into two sequences consisting of odd and even samples respectively. In addition, it was also shown that the aliasing introduced during the creation of these two sequences is eliminated when the two are recombined. Therefore, if one starts with two sequences, with the second offset half a sample period after the first, then it should be obvious to see how one could create a sequence of twice the sampling density by zero padding and then addition.

If this concept is extended into two dimensions then it follows that four aliased input sequences are required; odd rows/odd columns, odd rows/even columns, even rows/odd columns, and even rows/even columns. Thus one could construct a four camera rig with each camera offset by half a pel from its neighbors as shown in Figure 4.4.

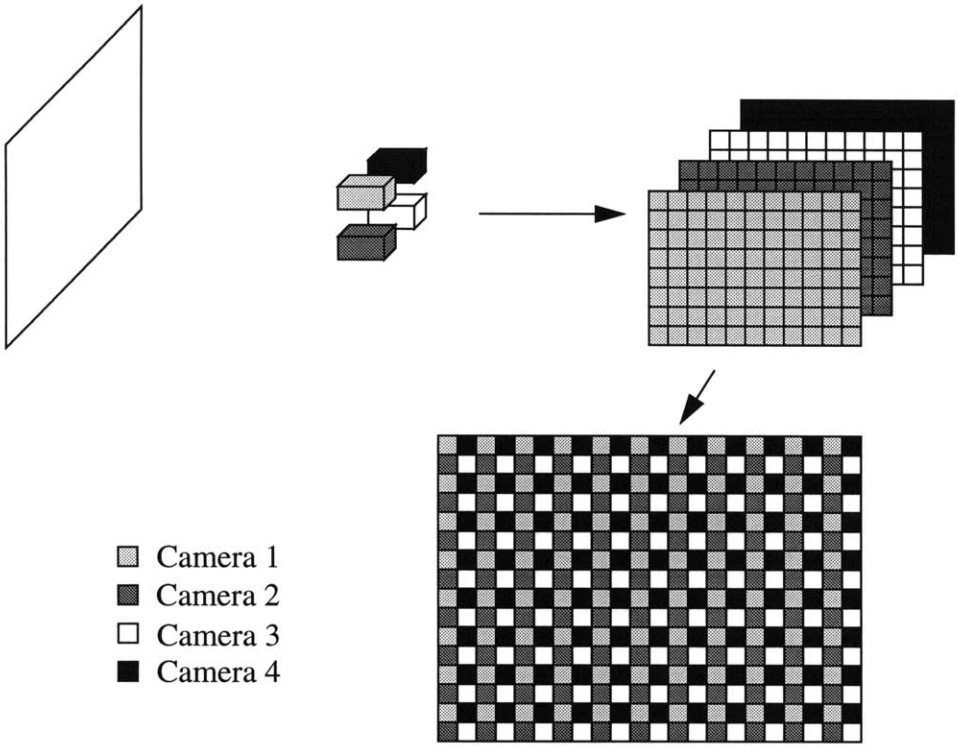


Figure 4.4 Four Camera Rig for Doubling Spatial Resolution

Generally, such a system will not result in an exact doubling of the spatial bandwidth as it relies on the assumption that frequency response of the camera is flat and contains aliasing of only the first harmonic as shown below,

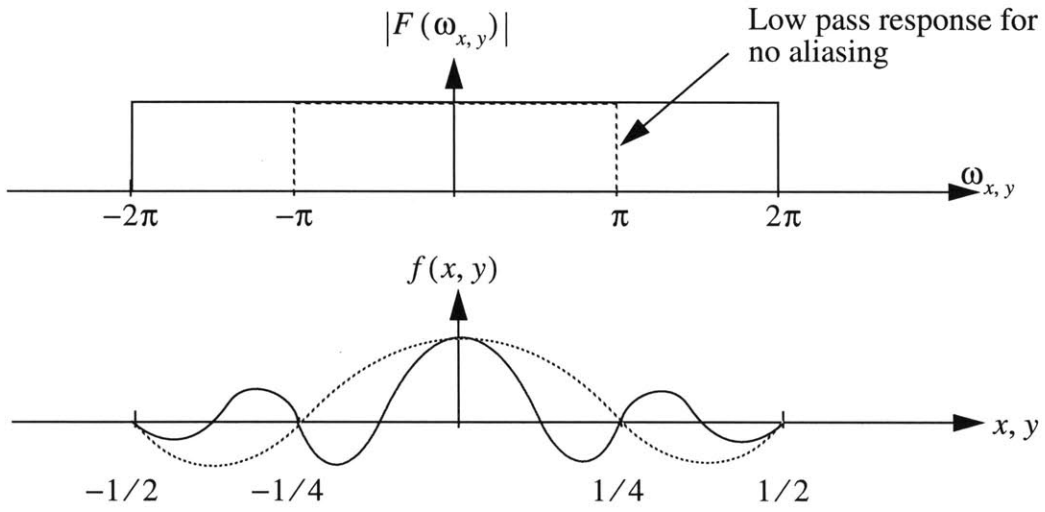


Figure 4.5 Desired Camera response for doubling resolution

Real cameras are incapable of generating signals with the response shown in Figure 4.5 for three reasons;

- First, camera optics are not perfect and therefore they introduce a certain amount of distortion into the image.
- Secondly, the sensors that form the camera chip are not infinitely small, they must have a significant finite area in order to reduce noise and also to minimize the magnitude of higher order aliased sidelobes.
- Third and finally, a real sensor is incapable of generating a negative response. Therefore it is impossible to implement $f(x, y)$ as the response must be that of an all-positive filter.

Having deduced that it is impossible to obtain the desired response from practical sensors, one should determine how close the response of a practical sensor is to this ideal. Consider the 9 element ccd array shown in Figure 4.6,

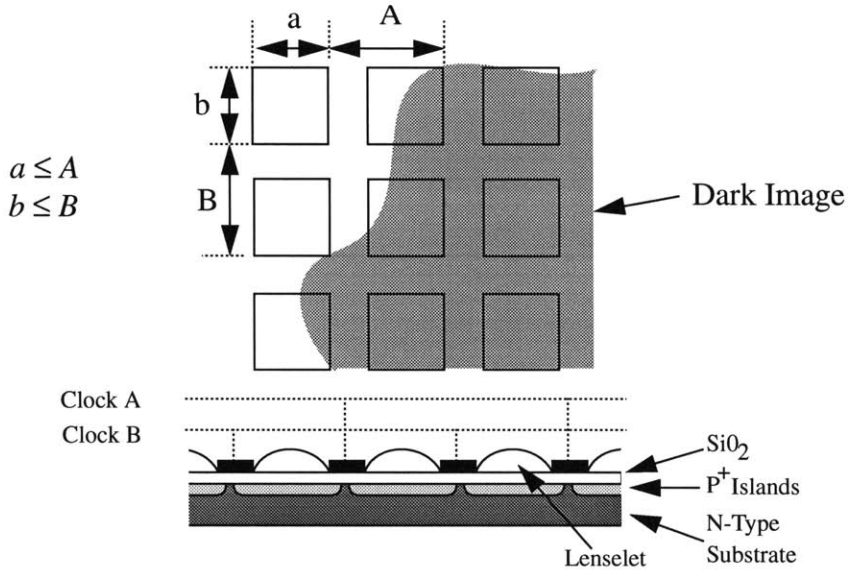


Figure 4.6 Camera CCD Array (modified from Fig 2.22 in [5])

The sensors in this array have dimension a by b and are distributed on a rectangular grid with cells of size A by B. A common measure used to describe the geometry of the sensor array is the fill factor:

$$\frac{ab}{AB} \tag{4.3}$$

If the effect of the lenselets and other camera optical elements is temporarily ignored, the frequency response due to the finite sensor area can be defined in terms of a separable function consisting of two sinc functions with nodal frequencies as shown in Figure 4.7. Notice that the response is low pass and that its width is dependent on the sensor size. Smaller sensors result in wider response than larger ones that filter out more of the higher frequencies.

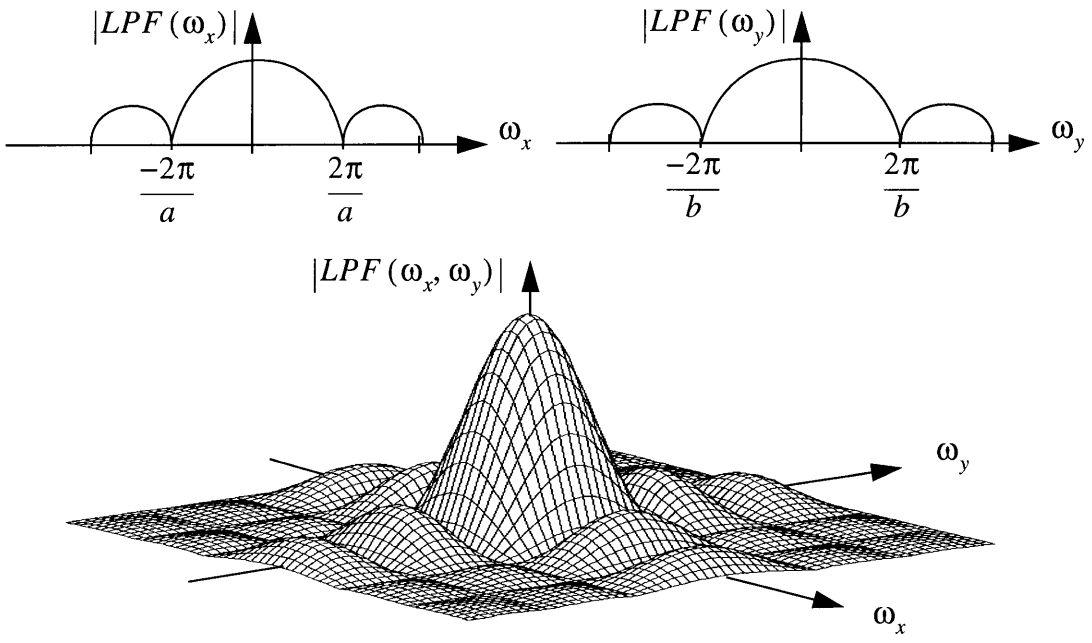


Figure 4.7 Low pass frequency response due to finite sensor element size

Reading the data from the ccd has the effect of multiplying the double-sinc frequency response shown in Figure 4.7 with the two dimensional pulse train shown in Figure 4.8,

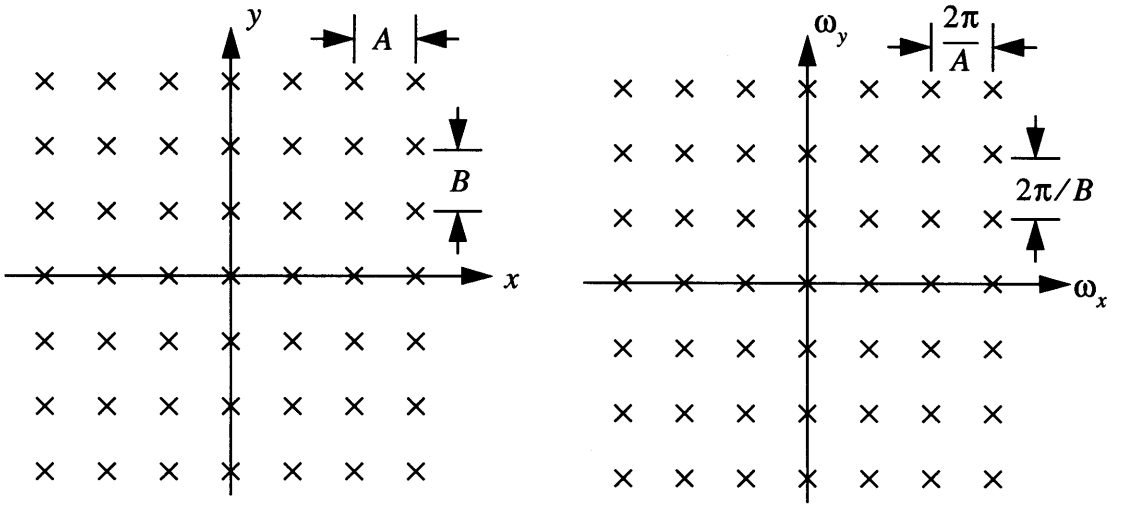


Figure 4.8 Two dimensional pulse train

When one combines the effects of all these operations on the projected image one obtains the Modulation Transfer Function (MTF) which describes the behavior of the recording system. Following on from the previous examples the MTF for the ccd array in Figure 4.6 is shown below (NB only the x axis projection is shown for clarity).

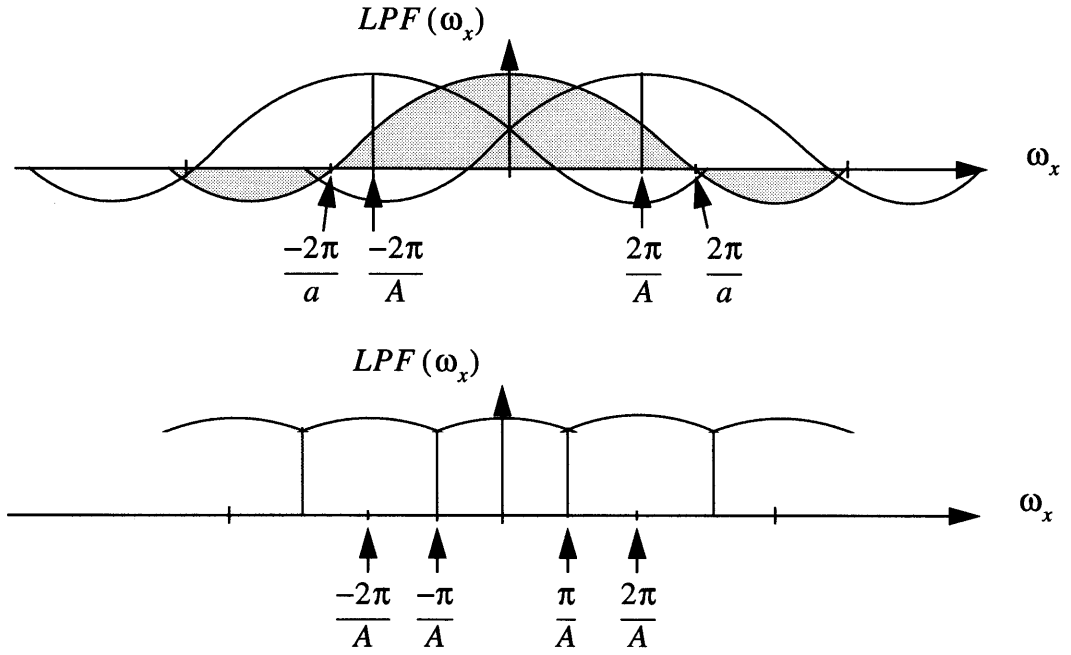


Figure 4.9 Modulation Transfer Function for a typical ccd array

At this stage one should note that there is significant aliasing in the sensor array's MTF. In fact, one wonders if it would be possible to obtain a recognizable image from the camera. However, one should remember that the MTF shown in Figure 4.9 was derived for the camera sensor array only. No consideration was given to the lenselets and camera lenses in its derivation. When one includes these optical elements into the system description, the end result is that the image projected onto the ccd array is filtered by a much narrower filter than that implied by the ccd array's geometry. Generally, most cameras are deliberately designed to be slightly unfocused in order to perform this type of pre-filtering. For more details on the effect of focus on MTF responses see [7].

Combining the optics, sensor array and sampling into a single entity results in the system shown in Figure 4.10. (N.B. only the x axis projection is shown in the interests of clarity)

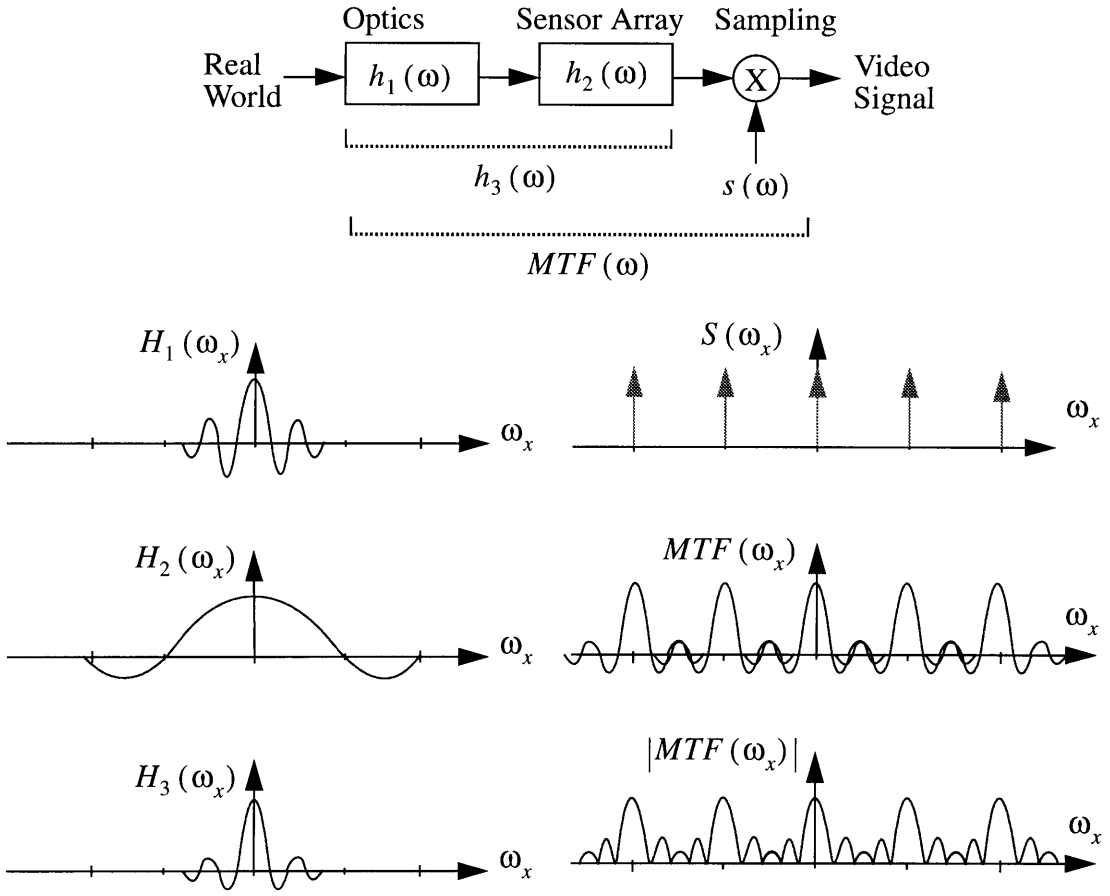


Figure 4.10 Modulation Transfer Function for a typical camera

When one compares the desired camera response shown in Figure 4.5 with that of the actual response shown in Figure 4.10 it becomes apparent that the degree to which the resolution can be enhanced through the use of this technique is somewhat limited. The critical part of the system is the optics. If the camera is perfectly focused then the optical response is flat and the resulting image will be too highly aliased in order to extract the lower harmonic spatial frequencies. Conversely, an out of focus camera will effectively low pass filter the image so that the higher spatial frequencies are virtually eliminated.

4.3 Enhancing Resolution Using One Camera

Good quality cameras are expensive, so it makes sense to find a way to achieve the same results using only a single camera. Two methods that only use one camera for resolution enhancement are described in this section.

4.3.1 Stepped Sensor Mount

The simplest method which is capable of achieving the same results as the four camera rig involves a movable camera sensor that steps to another camera position after capturing each frame. This process is repeated so that a frame is captured at each of the four positions in turn, thus generating a cyclic sequence with a periodicity of four frames. This method sacrifices temporal resolution for spatial resolution, which is not unreasonable considering the aim is to increase the resolution of the background; a near stationary region for which temporal resolution is not critical. However it is not perfect as it still suffers from two major flaws.

First, the input signal is assumed to be aliased as per the description in the previous section. If this condition is not met then no resolution enhancement will be possible as the spectra of the four images will be identical apart from the phase shifts caused by the different sample positions. The only gain possible will be in the form noise reduction from the multiple samples present.

Secondly, the mechanics or electronics involved in moving the camera sensor in the precise manner required would increase the cost significantly to the point where it may be cheaper just to use a camera already capable of the desired resolution. In fact it is highly likely that it would be impossible to achieve the exact repetitive motion desired. Thus, the

only way to ensure the half pel offsets is to use the higher resolution camera to begin with which once again defeats the purpose of this thesis.

4.3.2 Generalized Camera Motion

The obvious solution to this problem is to allow the camera to pan and zoom freely and then to motion compensate successive frames to fit one of the four camera positions. Apart from realizing the one camera goal, this method also affords another benefit in that the resolution can be increased in a somewhat less subtle but more robust method through the use of additional memory.

Consider that case in which the camera motion is purely translational and parallel to the background plane. In this case each image only requires a translation in order to align it with the appropriate sampling points and hence one has to rely on the odd/even sub nyquist sampling method described in the previous section in order to increase resolution. However, if zooms are allowed then the frequency response of the camera will be scaled during the motion compensation process. Perhaps, this can be best explained if one considers the scene below.

Figure 4.11 Two Shots of the Same Scene a) Wide angle, b) Close up (2 X)

For a given region there are four times as many pels (and hence twice the spectral resolution both horizontally and vertically) in the close up shot when compared to the wide angle shot. Therefore when one warps the wide angle image into the same scale as the close up and then examines their relative spectral content one finds that it is possible to reconstruct the portion of the image captured in the close up with a higher resolution than the wide angle image which is used to reconstruct the remainder of the image.

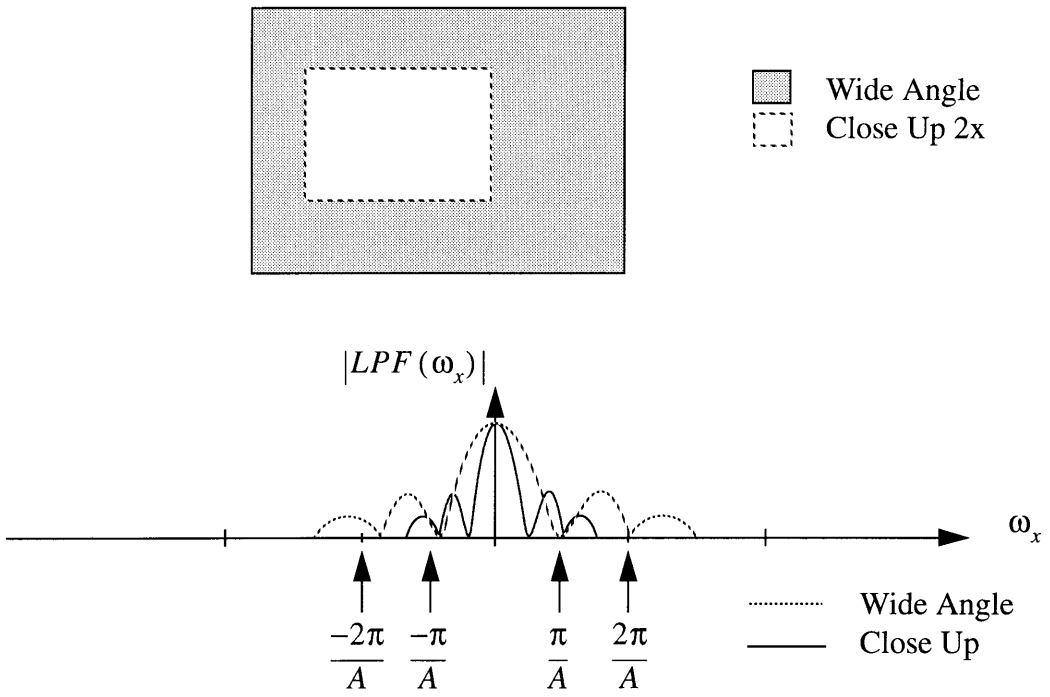


Figure 4.12 Spectral Content of two pictures when combined

The process of warping one image into the same scale as another, and thereby shrinking or expanding the range of its spectrum, forms the basis for the background resolution enhancement portion of the coder. Of course the degree by which the resolution is enhanced is highly dependent on the accuracy of the warp, and the means used to incorporate the information provided by each new frame into the representation. If the warp is not particularly accurate then the resolution may actually be decreased below that of one of

the original frames! Therefore some consideration should be given as to how to generate the most accurate warp possible and also how to update the prediction buffer to allow for the fact that warp will not necessarily be entirely accurate.

4.4 Enhanced Resolution Image Construction

The enhanced resolution coder constructs enhanced resolution images of the background from several previously transmitted images. Each of these images has a single six parameter affine warp model associated with it which describes the interframe motion between the model image and the current one. The affine model,

$$\begin{aligned} d_x &= a_x + b_x x + c_x y \\ d_y &= a_y + b_y x + c_y y \end{aligned} \tag{4.4}$$

has six parameters and accordingly the types of motion it can represent accurately is limited. In general the affine model will be accurate in situations where there is little 3D motion present. Rotations not about the optical axis or scenes containing significant perspective or depth in the image will not be modelled accurately. However, if one constrains the scene so that the background is roughly planar and perpendicular to the optical axis of the camera then the six parameter affine model is usually accurate enough to model the interframe motion.

In addition to the affine model, a coarse segmentation mask is also stored with each model to indicate which parts of the model are accurately described by the warp. Without this mask foreground objects may be incorrectly rendered into the background. Details of how the segmentation mask is derived are described in the next chapter.

Having determined the warp model, consideration must be given as to how to generate values that don't fall on original sample points. It is a well known fact that the best interpolation function for a band limited signal is one based on sinc functions. However, as the sinc function requires infinite support and hence infinite computation it is rarely used. Instead the computationally less expensive bicubic warp below is used,

$$\begin{aligned} xfrac &= x - xint & xint &= int(x) \\ yfrac &= y - yint & yint &= int(y) \end{aligned}$$

$$\begin{aligned} x_1 &= xfrac^2/2 - xfrac/3 - xfrac^3/6 & y_1 &= yfrac^2/2 - yfrac/3 - yfrac^3/6 \\ x_2 &= 1 - xfrac/2 + xfrac^3/2 & y_2 &= 1 - yfrac/2 + yfrac^3/2 \\ x_3 &= xfrac + xfrac^2 - xfrac^3/6 & y_3 &= yfrac + xfrac^2 - yfrac^3/6 \\ x_4 &= xfrac^3/6 - xfrac/6 & y_4 &= yfrac^3/6 - yfrac/6 \end{aligned}$$

$$t_1 = y_1 \begin{pmatrix} x_1 pred(xint-1, yint-1) + \\ x_2 pred(xint, yint-1) + \\ x_3 pred(xint+1, yint-1) + \\ x_4 pred(xint+2, yint-1) \end{pmatrix} \quad t_2 = y_2 \begin{pmatrix} x_1 pred(xint-1, yint) + \\ x_2 pred(xint, yint) + \\ x_3 pred(xint+1, yint) + \\ x_4 pred(xint+2, yint) \end{pmatrix}$$

$$t_3 = y_3 \begin{pmatrix} x_1 pred(xint-1, yint+1) + \\ x_2 pred(xint, yint+1) + \\ x_3 pred(xint+1, yint+1) + \\ x_4 pred(xint+2, yint+1) \end{pmatrix} \quad t_4 = y_4 \begin{pmatrix} x_1 pred(xint-1, yint+2) + \\ x_2 pred(xint, yint+2) + \\ x_3 pred(xint+1, yint+2) + \\ x_4 pred(xint+2, yint+2) \end{pmatrix}$$

$$newval = t_1 + t_2 + t_3 + t_4 \tag{4.5}$$

The support area of the bicubic warp above is the 4 by 4 region of pels centered on the position of the predicted pel (x, y).

The basic mechanism for constructing an image is very similar to that used by Irani and Peleg in their work on multiple motion analysis [16] and also that of Teodosio in her work on Salient Stills [28]. Each pel to be displayed is reconstructed from a number of previous frames, by collecting a candidate value from each model frame via a bicubic warp Eqn (4.5) and then evaluating a cost function to determine the best value. Unlike the Salient Still however, the models within the enhanced resolution coder also contain masking information which increases the accuracy of the reconstruction considerably as pels corresponding to foreground objects are no longer considered.

The Salient Still algorithm generally gives the best results when the pel's color is determined by evaluating a weighted median function, where the weight associated with each candidate value is dependent on the relative scale of the model that it comes from. In other words, models with small b_x and c_y values corresponding to close up images are favored over those which had large b_x and c_y values and hence correspond to wide angle images.

$$weight \propto \frac{2}{(2 + b_x + c_y)} \tag{4.6}$$

The enhanced resolution codec extends this concept one step further by including a measure of best fit to the sampling grid, the rationale being that the model providing the closest fit gives the most accurate prediction. The addition of this term generally only effects the outcome when images are of relatively similar scale, which makes sense as at other times a model of considerably smaller scale should have a considerably higher weight value than that for an image of large scale.

$$weight \propto \frac{1}{(b_x + c_y)\ (1+d)}$$

d = distance squared to nearest pel in model in terms of model scale, i.e by definition

$$0 \leq d \leq \frac{1}{2} \tag{4.7}$$

This is most easily shown in the diagram below,

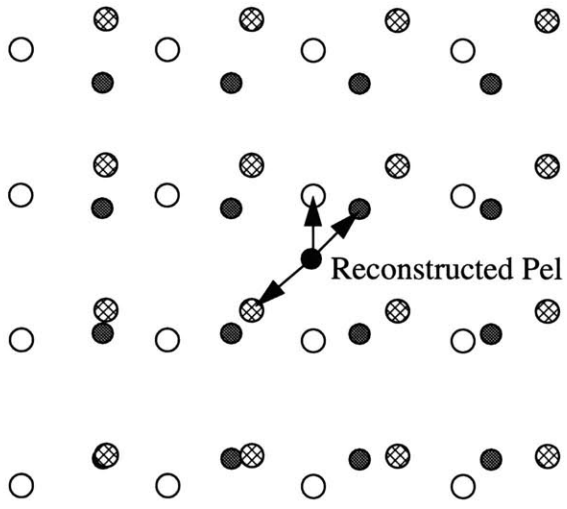


Figure 4.13 Nearest Model Pel Determination

Here, three models contribute candidate values to the median table. The scales of the models corresponding to the clear sample and cross hatched samples are the same. However, the shaded sample not only from a model of smaller scale but is closer to the reconstructed pels position. Therefore the candidate value corresponding to the shaded model will have the largest weighting.

Chapter 5

Foreground Extraction

The techniques of 6 parameter affine interframe motion representation and background resolution enhancement suffer from the assumption that no foreground objects are present in the input frames. Therefore, as any real or interesting footage will most likely contain foreground objects, a method must be found that either removes or *nearly* removes any foreground objects from the input. Once this is achieved the modified input can be used to generate a more accurate background motion model and hence a more accurate enhanced resolution background.

This chapter describes the development of an algorithm that extracts foreground regions from a frame. It should be noted that the method described herewith is not designed to perform perfect segmentation, as section 5.1 will show this is a near impossible task. The design criteria for the extraction algorithm is somewhat more relaxed. So long as the modified input contains no foreground information that will corrupt the background image under construction then the algorithm is deemed successful.

5.1 Choosing a Basis for Segmentation

The human visual system is extraordinarily good at performing the task of segmentation. Through the use of brightness, color, texture, lighting, motion, relative proximity, and stereo it can segment even the most complex scenes into a collection of related meaningful parts.

For example upon examination of Figure 5.1. below, one might segment the scene into the following list of objects; a flower bed, trees, sky, houses and lamp posts.

Figure 5.1. Flower Garden Scene (from an MPEG test sequence)

Even though this image is flat and presented in black and white, the segmentation described above still used four of the aforementioned methods for segmentation; brightness, texture, relative proximity, and shape. However, one other method not previously mentioned was also probably used -- *a priori* knowledge. This is extremely important as it means that people will have a bias or pre-conceived notion of how the segmentation should occur. In the context of the example above, most people will make use of prior experiences of what flowers, trees, houses and lamp posts look like, and therefore be predisposed to arriving at the conclusion that the tree is a foreground object.

Another valid solution to the segmentation of Figure 5.1. would be to say that everything but the tree is painted on a card and that the tree is actually a hole in the card though which one can see a wooden pattern. Although this second solution is somewhat contrived, it shows that apriori knowledge plays a significant role in the way that humans attempt to understand images.

Unfortunately, we have yet to learn how to program computers to make use of this sort of knowledge. The concept of 'tree-ness' or 'flower-ness', while extremely easy for us to comprehend, is something that is extremely difficult to describe to a computer. For this

reason, the majority of current segmentation techniques rely heavily on artificially contrived situations such as blue screens and already known or uncluttered backgrounds. Those that don't may work well some of the time but invariably fail under certain conditions that all too often appear in real footage.

Therefore, any algorithm that relies solely on an accurate segmentation in order to work is doomed to failure. Consequently, the algorithm described in the remainder of this chapter is designed to fail. Yes fail, so long as it doesn't allow foreground objects to corrupt the background it doesn't matter how much background is leaked into the foreground. Conversely, foreground objects that happen to be identical to the background can be leaked into the background as the end result will be the same. Of course this is rather glib but the reasoning is sound; an important goal of this research is to construct an enhanced resolution representation of the background of a scene. If perfect segmentation is not a prerequisite for this to be achievable then one should not aim for perfect segmentation!

One of the simplest methods for performing segmentation between a foreground object and a background is to use the interframe motion to segment on the basis of relative velocities between regions in the image. Although somewhat hidden, the majority of motion estimators rely heavily on the brightness, texture, and shape of the objects present in the image. Consequently making segmentation on the basis of interframe motion is not an unreasonable proposition as all these characteristics are automatically incorporated into the decision process. The following sections describe the fault tolerant motion based segmentation algorithm used to extract the foreground from the input images before they are used to update the background.

5.2 Motion Estimation

Current motion estimation methods all give false motion estimates under certain conditions; block matching fails where more than one motion is present in a block or where the luminance surface is smooth and optical flow fails badly at motion boundaries. However, with this understanding it is possible to take one of these algorithms and modify its operation to note which regions are likely to generate false motion estimates and to subsequently minimize the presence of false motion estimates in these areas. This section will briefly describe several modifications made to a standard block matching algorithm in order to improve its performance in two areas.

First, the amount of computational complexity is reduced through the use of coarse to fine hierarchical pyramid techniques that generate an initial coarse estimate of the motion using low resolution images and then subsequently refine the motion estimates using images of successively higher resolution.

Secondly, the presence of false motion estimates resulting from blocks containing more than one motion or comprised of flat luminance data is minimized by;

- Making the block size a maximum of 8 by 8 pixels in areas of non uniform motion.
- Using a spiral search originating at the vector corresponding to zero motion
- Weighting the error criterion so that vectors closer to the zero motion vector are preferred over those at the extremes of the search range. The degree to which low magnitude motion vectors are preferred over high magnitude vectors is dependent on how flat the block being examined.

- After each stage of refinement in the hierarchy the motion estimates outliers resulting from noise on flat blocks were filtered using a median filter.

More explicit details of the motion estimation algorithm are given in Appendix A.

5.3 Background Model Construction

As was shown in Chapter 4, finding an appropriate model for the background of an image scene is not an easy task. However, if one is willing to make a few, admittedly non-trivial, assumptions then it is possible to simplify this task considerably. The assumptions made in order for the enhanced resolution coder to work are;

- The projected area of the background on the images presented to the encoder is large relative to that taken up by foreground objects.
- The distortion introduced by the camera optics is minimized through the use of long lenses and a restricted field of view.

Having made these assumptions, in particular the second, then one can represent inter-frame background motion using the 6 parameter model,

$$\begin{aligned} d_x &= a_x + b_x x + c_x y \\ d_y &= a_y + b_y x + c_y y \end{aligned} \tag{5.1}$$

Using the sparse set of motion estimates generated by the motion estimator in Appendix A this model is derived using the least squares algorithm in Appendix B.

5.4 Foreground Model Construction

Having obtained a reasonably accurate model for the background motion it becomes necessary to determine where it fails and hence which regions of the image should be transmitted as foreground information.

5.4.1 Initial Segmentation

Using the least squares plane fitting algorithm described in Appendix B does more than just generate the 6 parameter affine model for the background motion. It also generates an initial segmentation based on the magnitude and direction of the motion vectors. This occurs as the outliers must be removed from the input data in order to generate an accurate model by the least squares method. The set of blocks corresponding to outlier motion vectors generated by the plane fitting algorithm forms the initial segmentation.

5.4.2 Variance Based Segmentation Adjustment

As mentioned earlier in this chapter the output from the motion estimator cannot be relied upon for accuracy, even when the modifications described in section 5.2 are incorporated into its operation. Therefore, it is highly likely that some blocks will be tagged as foreground regions even though they are clearly part of the background. The segmentation adjustment procedure described in the following paragraphs attempts to identify these blocks and reassign them to the background. However, the procedure does more than that as it also allows currently tagged foreground blocks to become background if the resulting error from using the background motion is no worse than if the foreground motion was used.

As might be expected the rationale for making the adjustment lies in the knowledge that the motion estimator is likely to fail for blocks that are flat. Such blocks are character-

ized quantitatively by having low variance. These are the only blocks which are candidates for adjustment, which is performed in two phases; reassignment and filtering.

The reassignment phase is rather simple. The variance of each block (having already been calculated during motion estimation) is compared to a threshold value dependent upon the blocks size. If a block's variance falls below the threshold then the block is a candidate for readjustment. Residuals from using the modelled background motion and also the foreground motion are then computed for candidate blocks. If the background residual is smaller or only slightly larger than that for the case when the foreground motion is used then the block is reassigned to background.

The second filtering phase attempts to remove salt and pepper noise from the segmentation by filtering with a 3 by 3 median filter.

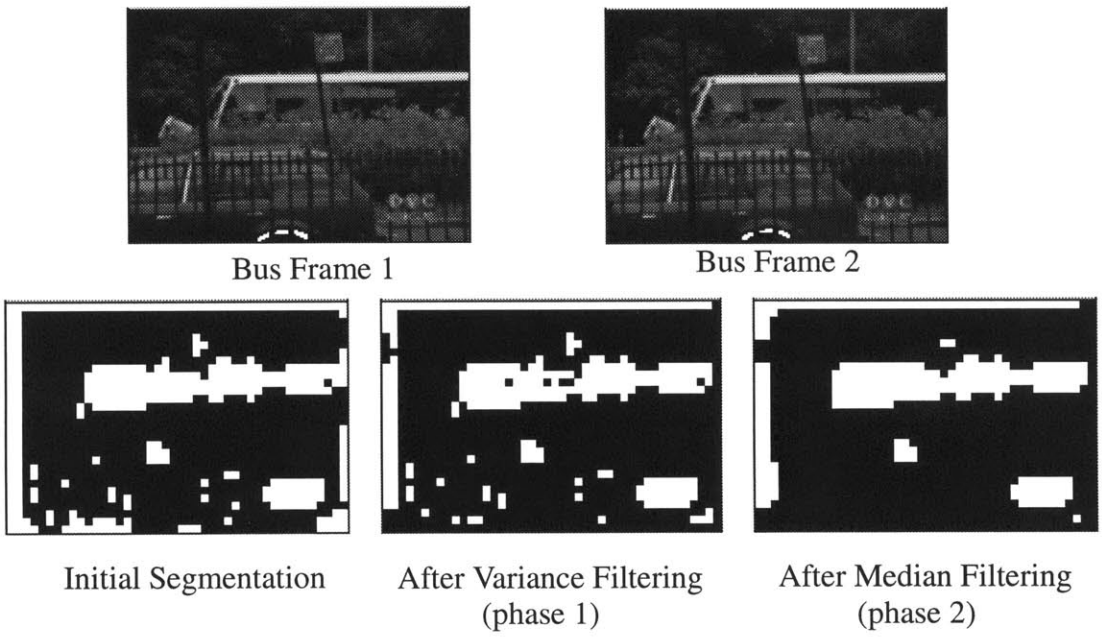


Figure 5.2 Variance Based Adjustment

5.4.3 Region Allocation

Having appropriately filtered the segmentation on a blockwise basis, a more sophisticated filtering is performed which takes the blocks' connectivity into account. The first part of process consists of a scan line conversion algorithm which scans the segmentation mask in a raster like fashion and tags each block as either background or as part of a foreground object. After the scanline conversion process notes but does not merge new objects with previous ones to which they are connected, a second pass is performed after scanline conversion which merges equivalent regions.

The output from this process is a segmentation map with each block tagged as background model compliant or as one of a small number of regions.

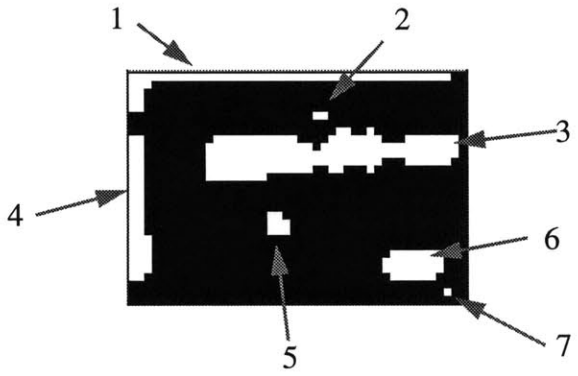


Figure 5.3 Example of region allocation by scan line conversion and merge

5.4.4 Region Pruning

The final stage of the segmentation process prunes regions from the segmentation according to their size and the total error resulting from using the coarse block based foreground prediction over the warped affine background model. Extremely small regions are assumed to be the result of noise, either in the motion estimator output or the input image. In these cases the added expense of transmitting small regions over the larger error signal

resulting from background model prediction is deemed unjustifiable and they are automatically reassigned to being part of the background.

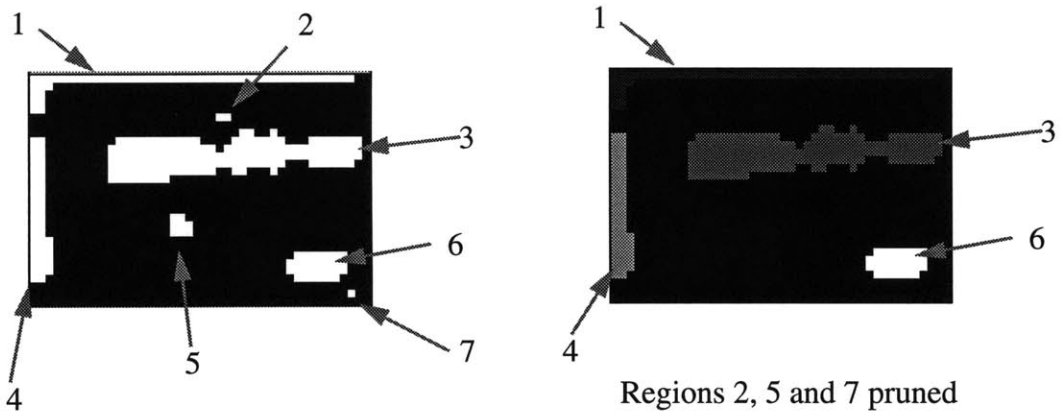


Figure 5.4 Example Region pruning

Thus after segmentation the following information exists about the current frame;

- a 6 parameter affine model which describes the background motion relative to the previous frame.
- a segmentation mask which indicates whether a block belongs to the background or to a numbered region. If the block does belong to a region then the number of that region is stored in the segmentation mask.
- a motion vector field which is used to describe the motion of each region on a block by block basis.

This information is necessary and sufficient to describe the foreground object, details of how it is encoded into the bitstream may be found in Appendix B.

5.5 Temporarily Stationary Objects

One of the major downfalls of the foreground model construction technique just described is that it depends on interframe motion being present between every frame pair presented to the encoder. Thus, any moving object that becomes stationary for one or more frames is then immediately and erroneously merged into the background. While this has little effect on the reconstruction of the current frame, it does significantly compromise the segmentation mask for that frame which may result in an inaccurate background construction for later frames.

A partial solution to this problem which makes use of first order proximity was implemented. The basis of the solution was to attempt to predict the position of foreground regions in the next frame from the position and motion data generated for the current frame. A "holding delay" was then introduced for the regions that were predicted to contain foreground information. Foreground regions whose motion matched that of the background for the duration of this delay were then reassigned to the background. If a region's motion did not match that of the background then its holding delay was reset.

As was stated, this solution was only partially successful. Details of how and why it failed for particular situations are presented in Chapter 7, Simulation Results.

Chapter 6

Updating the Models

The decision of how to update the various models within the coder/decoder is an important one. If the models associated with each frame are always dependent on those of previously received frames then several problems arise. First, it is impossible to "tune in" to the encoded stream at any other point but the beginning. Second, if an error occurs in the bitstream then recovery is impossible. The other extreme is to re-transmit each model in its entirety with every frame, which of course requires a higher bit rate than necessary. Clearly the best solution lies somewhere in between these two extremes.

Single frames of video form the models used within MPEG [3] which solves the problem of updating them through the use of three frame types; Intra (I), Predicted (P) and Bidirectionally predicted (B). Intra frames are independent of any other, while Predicted frames are predicted from the previously transmitted I or P frame and Bipredicitive frames are predicted from the two most previously transmitted I or P frames. As I frames can be decoded without prediction from any other frame, they are transmitted on a regular basis to allow for random access and also to ensure any transmission errors are flushed from the decoder buffers. Figure 6.1. below shows a typical MPEG frame allocation, the arrows indicate predictive coding.

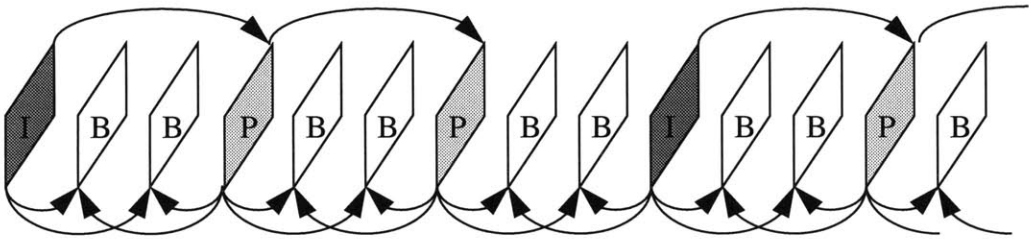


Figure 6.1. Example of MPEG frame ordering

6.1 Frame Types and Ordering

Like H.261 and MPEG, the Enhanced Resolution Coder encodes frames of video in several different ways in order to allow for random access and error recovery. Two distinct frame types are used;

- Model (M) frame
- Model Delta (D) frame

6.1.1 Model (M) Frames

The first frame to be sent is always a model or M frame. The M frame is essentially the same as an MPEG I frame in both nature and implementation, that is, it contains Huffman encoded quantized DCT coefficients but no motion information. It can, however, contain segmentation information if it is available. As this information is not available for the first frame of a sequence; the first frame never contains segmentation information.

M frames transmitted with segmentation information can be used to predict occluded information in subsequent frames; however if an M frame contains no segmentation information then it is only used to predict the following frame before it is discarded.

This feature is particularly useful for "pre loading" the background in video conferencing situations. The M frame also provides an avenue for the encoder to take advantage of other more computationally expensive algorithms such as Salient Still [28] or Layered Coder [4]. As these algorithms work on significantly more frames than can be stored in the enhanced resolution encoder, they may be able to generate a more accurate background representation. The ability to make use of the output from these algorithms may afford improved performance by circumventing the memory and computation bounds present in the standard enhanced resolution encoder.

6.1.2 Model Delta (D) Frame

The second frame type is the D frame. D frames are not unlike the MPEG B frames, in that they contain interframe motion information from one or more predictors, and an error signal consisting of Huffman encoded quantized DCT coefficients. However, unlike the MPEG B frame they also contain segmentation information about the position of a finite number of foreground regions. The background and foreground motion models transmitted within a D frame are complete and fully describe the motion and shape information for the frame, relative to the previous frame.

However the error residual is based on the composition process which uses many frames to construct the background prediction. Consequently, some discrepancies will arise if a decoder "tunes" in halfway through without having the previous background models to aid in the prediction. Therefore the decoder should not attempt to decode images until it detects an M frame in the bitstream.

In short the information contained in each frame can be summarized as follows

Frame Type	Background Motion	Foreground Motion	Foreground Shape	Error Residual
M	y or n	n	y or n	y
D	y	y	y	y

TABLE 6.1 Frame Types and their information content

6.1.3 Frame Ordering Example

An example of frame ordering in a typical sequence is shown below in Figure 6.2. Several things should be learned from this example. First, D frames always construct predictions using the previous frame for foreground objects and the previous frame plus any valid models for the remaining background. Secondly, frame 9, an M frame, contains a segmentation mask and can, therefore, be used as a model whereas the first M frame (frame 0) did not and had to be discarded after being used to predict frame 1.

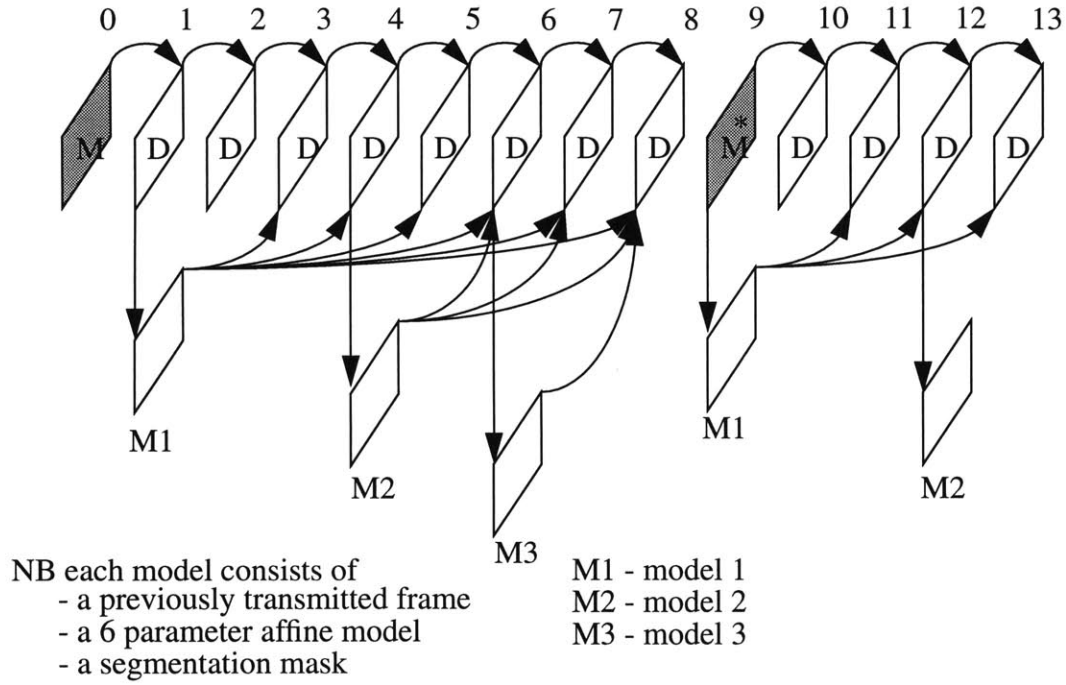


Figure 6.2 Enhanced Resolution Codec Frame Ordering Example

6.2 Updating the Affine Model

Adjusting the affine parameters of existing models to reflect motion between the previous frame and the frame just being encoded is crucial to the operation of the algorithm. Fortunately, affine models can be added according to the equations below [28] to give a single affine model that combines the effects of warping using model 1 followed by second warp using model 2

$$\begin{aligned} ax_{new} &= ax_1 + ax_2 + cx_1 \times ay_2 + bx_1 \times ax_2 \\ ay_{new} &= ay_1 + ay_2 + cy_1 \times ay_2 + by_1 \times ax_2 \\ bx_{new} &= bx_1 + bx_2 + cx_1 \times by_2 + bx_1 \times bx_2 \\ by_{new} &= by_1 + by_2 + cx_1 \times by_2 + by_1 \times bx_2 \\ cx_{new} &= cx_1 + cx_2 + cx_1 \times cy_2 + bx_1 \times cx_2 \\ cy_{new} &= cy_1 + cy_2 + cx_1 \times cy_2 + by_1 \times cx_2 \end{aligned} \tag{6.1}$$

This property of the affine model means that each affine model needs only to be transmitted once. This negates the need to perform motion estimation on frames other than the previous one. Subsequent warps are then taken into account by adding the current affine model to those associated with previous models.

Transmission of the affine model implies that some quantization of its parameters must take place. If too few bits are assigned to the parameters then the effects of quantization noise will quickly accumulate making the results of the summation progressively more inaccurate. Therefore, care must be taken to ensure that a sufficient number of bits are assigned to each of the parameters. For details on how the parameters are encoded see Appendix C.

6.3 Model Update Decision

Deciding which frames should be used as models for prediction of subsequent frames is critical to the performance of the encoder.

The simplest strategy is to cycle through the buffers throwing out the oldest model and replacing it with the current one. Unfortunately the amount of memory and computation required to implement a reasonable cycle length would make both the encoder and decoder prohibitively expensive. A more realistic approach would be to limit the number of models to perhaps somewhere between 4 and 8. Given a limited number of models two update strategies come to mind.

Consider the situation where instead of using every frame as a model, only every m^{th} frame is used where m is a small number, between say 2 and 5. This modification extends the temporal span (in terms of the time between the various frames used to construct the background) of the encoder by a factor of m. However, it still does not guarantee optimal model assignment in terms of maximal revealed background area. Another problem with this method is that it does not automatically take the scale of the frames into account, close ups are just as likely to be discarded as wide angle frames.

The second strategy addresses the shortcomings of the first. By performing a weighted correlation of the coverage provided by all but one of the currently valid models, including the current frame, it is possible to determine which model should be discarded so that maximal coverage was achieved for the given number of models allowed. The correlation process is done in two phases. First, the bounds of the area that could be covered by the set of models being evaluated is calculated. Next, a correlation is performed by summing the weight corresponding to the "best" model for each pel within the bounds of the reconstruction space. The weight factor shown in (6.2) attempts to find the most accurate recon-

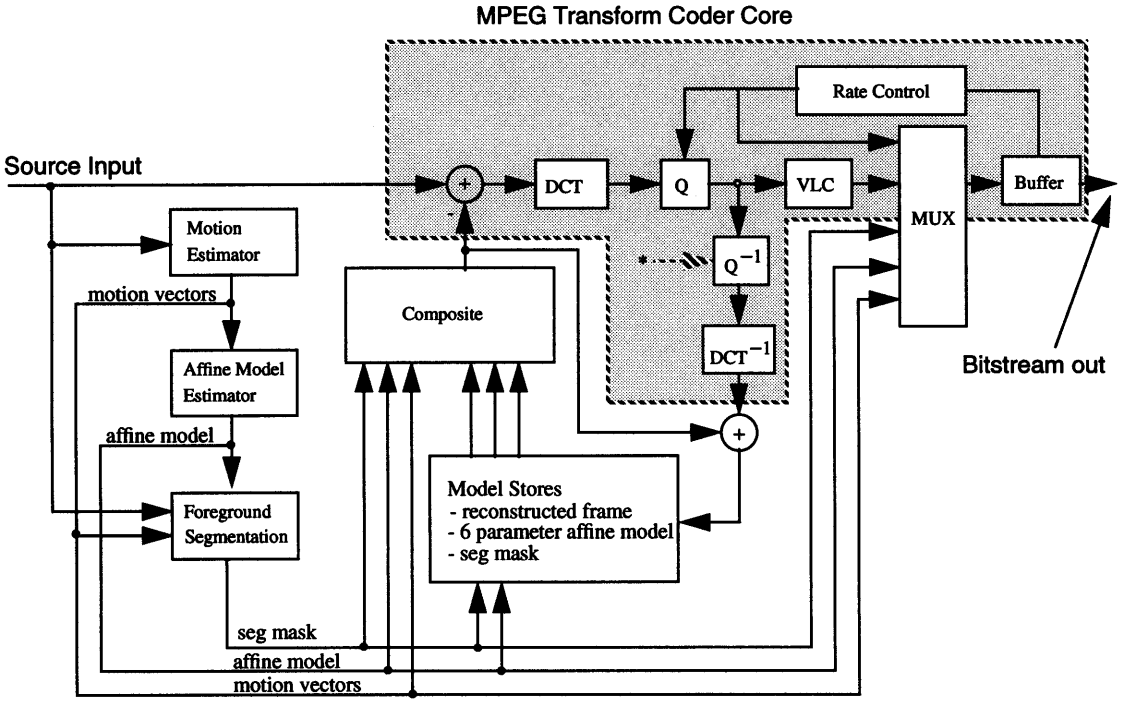
struction for a given pel. In addition, it takes scale into account so that the final correlation value corresponds to the total number of valid samples that can be used to reconstruct the background.

$$\frac{1}{(1+\frac{(b_x+c_y)}{2})^2} \tag{6.2}$$

This correlation is performed for each model in turn by omitting it from the set of candidate models, this subset is then used to generate the correlation value. The model whose omittance results in the highest correlation is replaced by the current frame. Note that this can result in the current frame being replaced by itself which corresponds to the case where the current models already provide the best coverage.

6.4 Encoder Block Diagram

Below is the encoder block diagram. Notice that a significant portion of the encoder, the transform coder core, is borrowed from the MPEG coder. Thus, the enhanced resolution coder does not require a complete redesign of an existing encoder, only modifications to support the new motion analysis and compositing sections.



DCT - 8x8 Discrete Cosine Transform
DCT^{-1} - 8x8 Inverse Discrete Cosine Transform
* quantizer step size

Q - Quantization
Q^{-1} - Inverse Quantization
VLC - Variable Length Coding

Figure 6.3 Enhanced Resolution Encoder Block Diagram

6.5 Decoder Block Diagram

The corresponding enhanced resolution decoder can be somewhat more complicated than its MPEG counterpart. The viewer is given a choice between viewing the standard decoded display or an enhanced display that takes advantage of the extra models stored in the decoder to construct a super resolution, screen widened version of the same image. If the super resolution image is not required then that part of the decoder can be disabled, thus reducing the computational complexity, and hence the cost of the decoder.

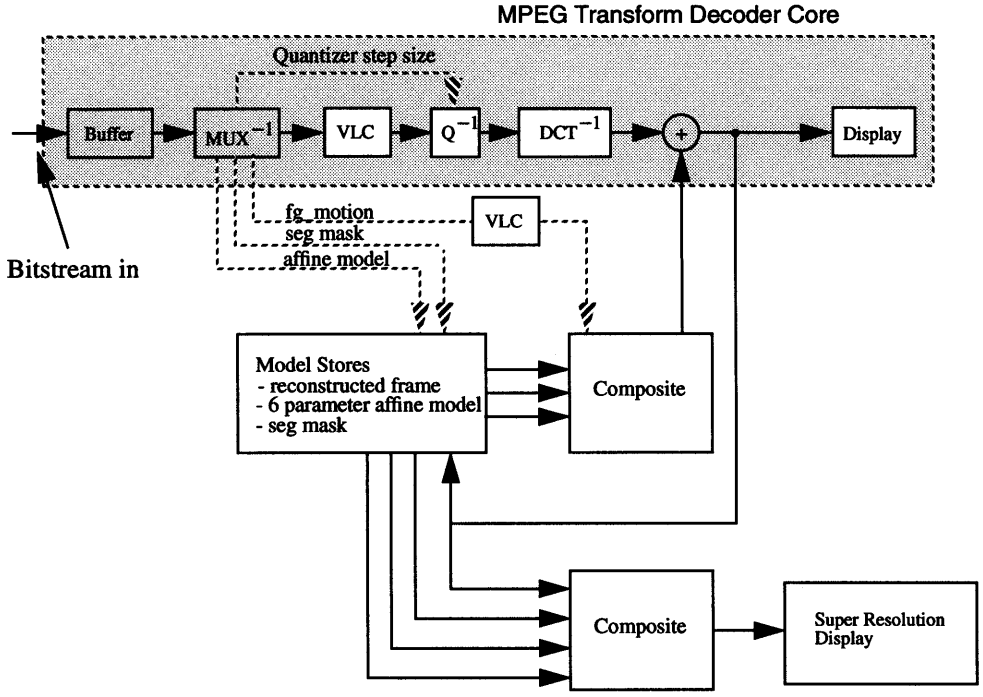


Figure 6.4 Enhanced Resolution Decoder Block Diagram

Chapter 7

Simulation Results

Two sequences were used to test the enhanced resolution codec.

The first sequence consisted of a short lecture presentation in front of a white board. Specifically designed to match the background model assumed by the enhanced resolution coder, it was hoped that the encoder would achieve a high performance for encoding this scene. This sequence consisted of 300 frames at a resolution of 352 pixels wide by 240 pixels high, 4:2:0 chrominance format (chrominance components sub-sampled by a factor of 2 horizontally and vertically relative to luminance component) and was coded at a target rate of 600kbit/s in order to achieve an average SNR between 30 and 35 dB.

As the first sequence was a "best case", the second sequence was chosen to represent the "worst case", a sequence containing significant three dimensional rotational motion in which the foreground dominates the viewed area. To that end, a sequence containing a close up of a revolving carousel was chosen. This sequence consisted of 150 frames at a resolution of 352 pixels wide by 240 pixels high, 4:2:0 chrominance format and was coded at a target rate of 1.2Mbit/s in order to achieve an average SNR between 30 and 35 dB.

The number of model frames in the enhanced resolution encoder was limited to eight. Also, motion estimation was performed as described in Appendix A to half pel accuracy and the segmentation was performed using a block size of 8 by 8 pels.

The MPEG encoder used for comparison in section 7.2 was developed in the Media Laboratory's Information & Entertainment Systems group. The MPEG encoder uses MPEG2 Test Model 5 rate control and an exhaustive search followed by a half pel refinement for motion estimation.

7.1 Motion/Shape Estimation Algorithm Performance

7.1.1 Foreground Object Extraction

Even though the motion disparity between the background and the "guest lecturer" was small, the encoder managed to segment the majority of the lecturer from the background. As was expected the segmentation was not perfect. In particular, regions with insufficient texture, e.g. the lecturers hair and arm, were initially incorrectly attributed to the background as a result of the motion estimator failing to detect the true motion for these regions.

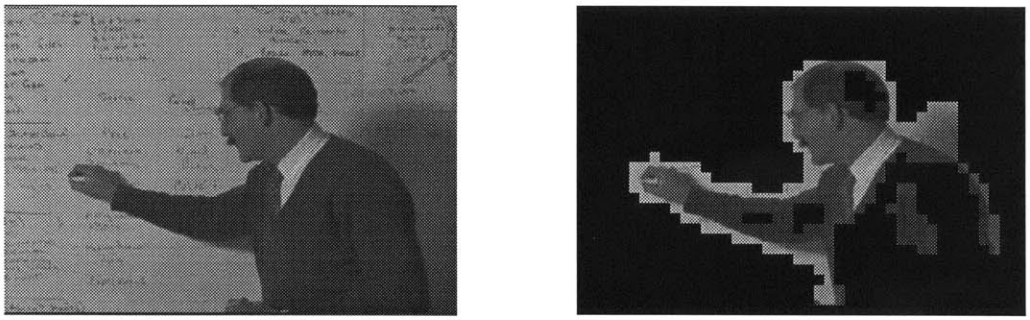


Figure 7.1 Original image and Segmented Lecturer with hole in head (and also in arm)

Another problem arose during pans when new parts of the background were revealed. Initially, these new parts were correctly tagged as a foreground but incorrect motion estimates due to lack of texture caused them to remain as foreground instead of becoming background. While this was not a problem for situations when the camera was only performing pans, problems did arise when zooms were included.

As closure is only guaranteed for the transmitted picture area, holes sometimes appeared in the supersized picture outside the transmitted picture area due to the fact that previous segmentation errors resulted in holes in the background. This situation was exacerbated by the fact that the foreground data was then subject to the "holding delay" put in place to prevent temporarily stationary foreground objects (see section 5.5) from being sent to the background.

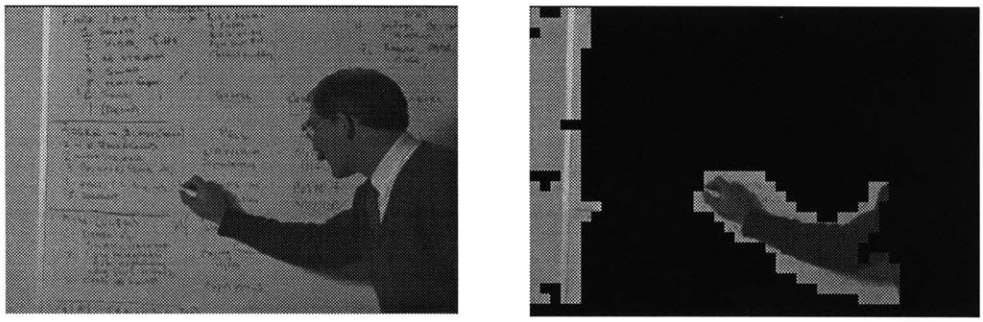


Figure 7.2 Poor segmentation due to flat regions causing inaccurate motion estimation

Apart from these two flaws, for which solutions will be suggested in the next chapter, the segmentation algorithm worked reasonably well for the "lecturer" sequence. However the same cannot be said for the carousel sequence. Due to the large amount of 3D motion in this sequence practically the entire frame was tagged as foreground.

Figure 7.3 Carousel Foreground Segmentation

7.1.2 Background Construction Accuracy

As might be expected, the enhanced resolution encoder was unable to construct a background of any worth for the carousel sequence due to the excessive 3D motion present.

The lecturer sequence was another story, however. Despite the coarse sampling of the motion vector field, the algorithm was usually able to generate reasonably accurate affine models for the sequence. In fact, when compared to the background at the original resolution, the background reconstruction was generally better due to the affine warp's capability to represent zooms. Of course the lecturer sequence was chosen with these ideas in mind, so it is hardly surprising that the algorithm should generate accurate models for its representation.

Figure 7.4 on the following page shows the super resolution background constructed by the decoder at the beginning, middle and end of a 90 frame Group of Pictures (frames 1, 51, and 89).

Initially the best estimate for the background left significant holes in the representation as shown by the black areas. By frame 51 the lecturer's arm had been removed from the representation along with most of his head. However, by frame 89 the lecturer's head and torso had reappeared in the background. The reason for this is that the encoder was set to only assume a maximum of eight model stores in the decoder. Therefore, only the "best" eight frames were kept. These eight frames provided sufficient information to reveal the white board panned off to the right and the region under the lecturer's arm, but in doing so was unable to reveal the regions behind the lecturers head and torso.

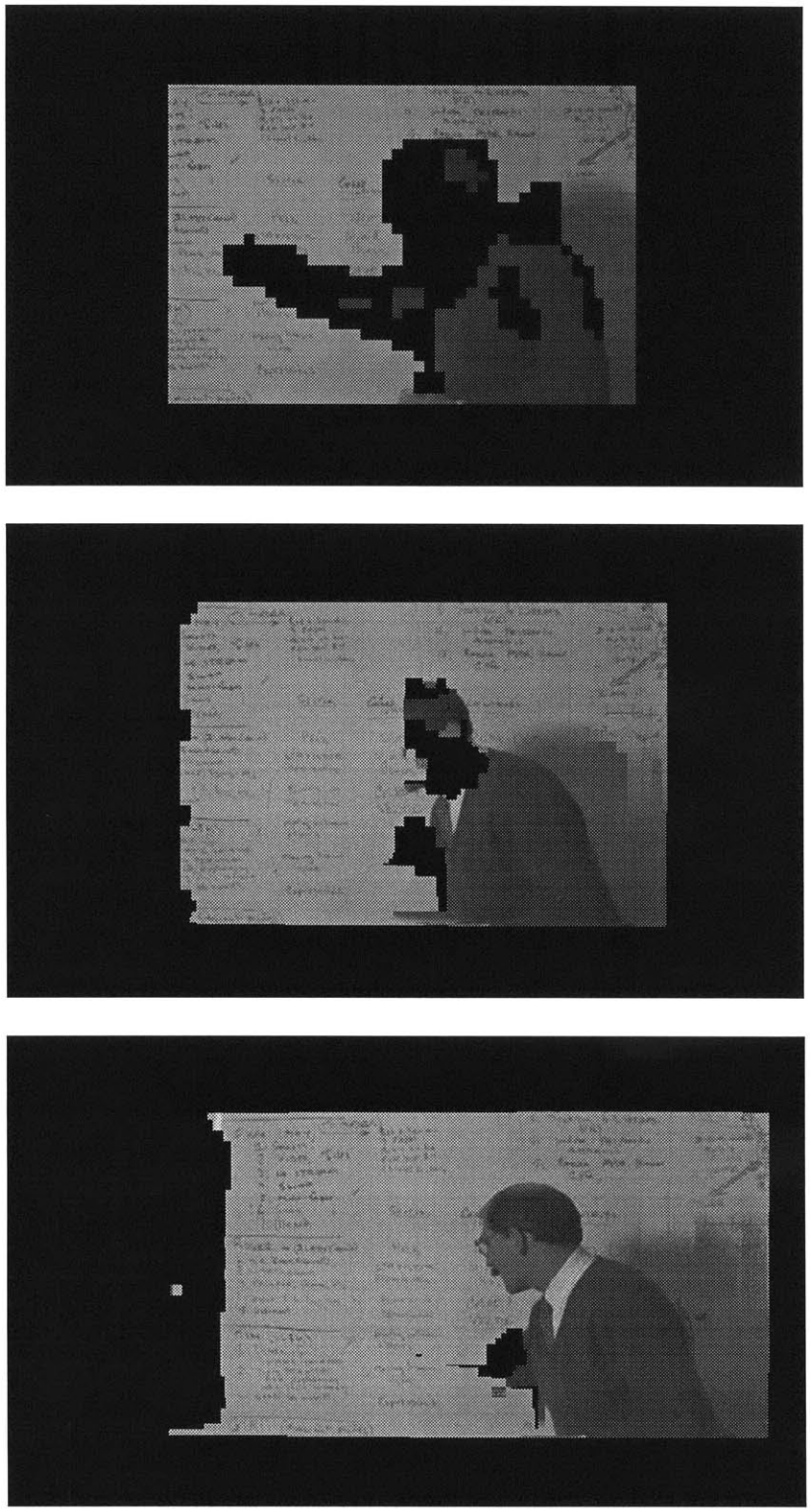


Figure 7.4 Super Resolution Backgrounds for frames 1, 51, and 89

7.1.3 Super Resolution Composited Image

In addition to generating a super resolution background, the decoder also has the capability to generate a second super resolution image that also includes the transient foreground information from the current frame. Below is an example of frame 89 with the foreground composited over the super resolution background buffer.

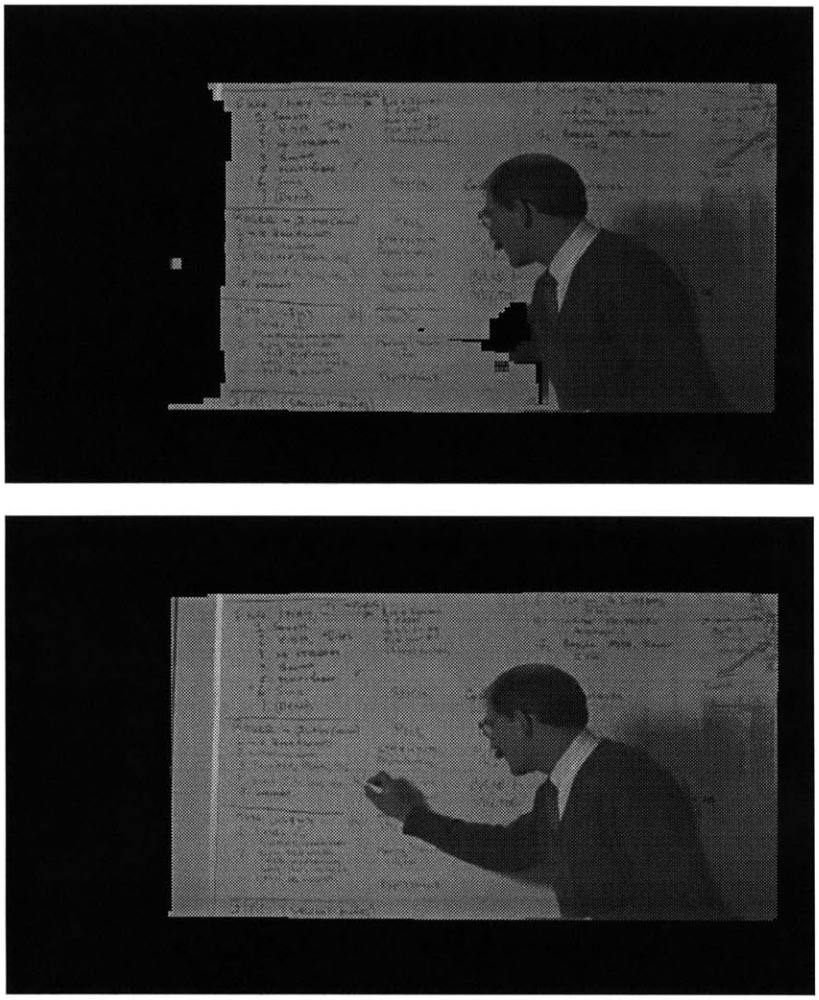


Figure 7.5 Super Resolution Frame with foreground composited on top

Notice that holes in the background caused by segmentation errors are concealed by the foreground for the region covered by the transmitted frame.

7.2 Efficiency of Representation compared to MPEG2

In order to make the comparison as fair as possible, both the MPEG and enhanced resolution algorithm parameters were matched. To that end the bit rates were set equal and the I frame Rate of MPEG was set to be the same rate as that for M frames in the enhanced resolution coder. The lecturer sequence was coded at 600kbit/s while the carousel sequence was coded at 1.2Mbit/s.

7.2.1 Motion/Shape Information

As can be seen from the plot below for the lecturer sequence, the number of bits spent by the enhanced resolution coder on motion and shape information was significantly less than that spent by the MPEG1 algorithm for motion alone. MPEG spent an average of 3614 bits/frame on motion while the enhanced resolution coder spent an average of 2256 bits/frame, a savings of 1358 bits /frame or 38%.

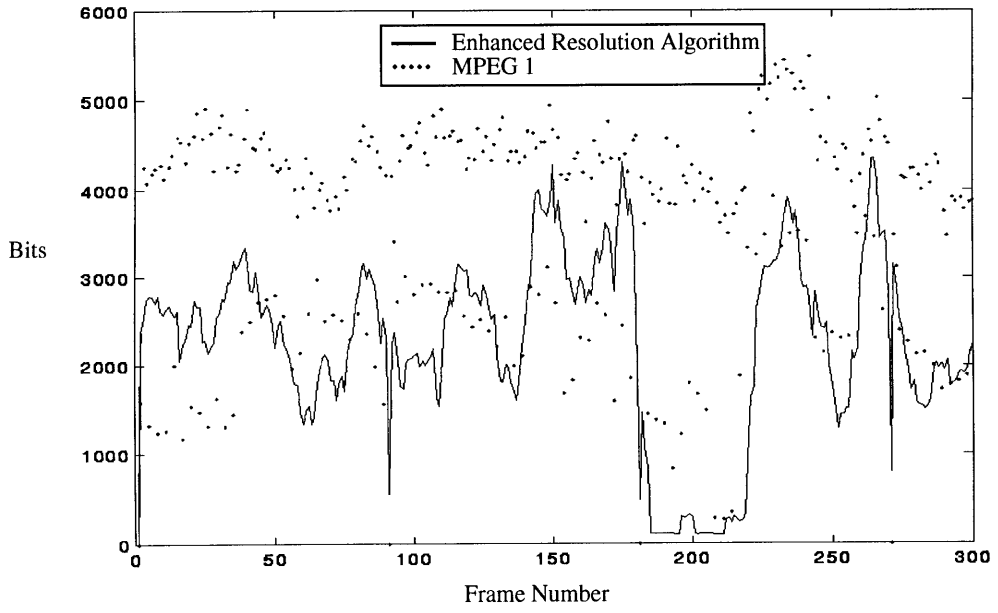


Figure 7.6 Motion Information bit rates for the Lecturer Sequence

The tables were turned, however for the carousel sequence. Due to the overwhelming amount of foreground information, the enhanced resolution encoder used nearly twice as many bits as the MPEG encoder to encode the motion. However, when one considers the fact that the enhanced resolution coder used blocks of size 8 by 8 pixels instead of 16 by 16 pixels, and hence had four times as many motion vectors to send, the factor of two increase in bits used does not look so bad after all.

If one ignores the differing block sizes, one could say that the enhanced resolution coder had fallen back into an MPEG like block based coder and as such should have perform liked one.

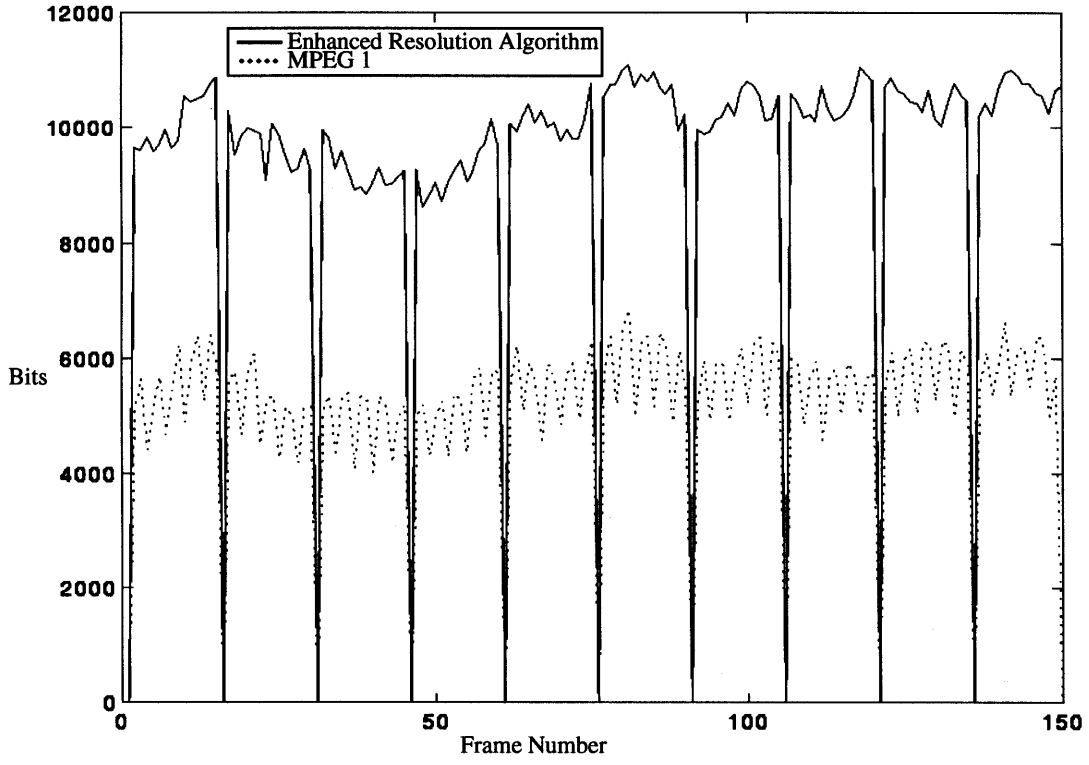


Figure 7.7 Motion Information bit rates for the Carousel Sequence

7.2.2 Error Signal

As can be seen from the plot below, the enhanced resolution algorithm performed considerably better in terms of SNR than MPEG for the lecturer sequence. This can be attributed to two main things. First, the motion representation was not only more accurate but also more compact than that for MPEG. This resulted in more accurate predictions but also allowed more bits to be used for the error signal. Second, the presence of multiple reference stores reduced the amount of information that had to be retransmitted due to cyclic occlusion.

Over the three hundred frame sequence the average SNR for the MPEG 1 coder was 33.15 dB while the enhanced resolution encoder achieved an SNR of 34.38 dB, an improvement of 1.23 dB over MPEG 1.

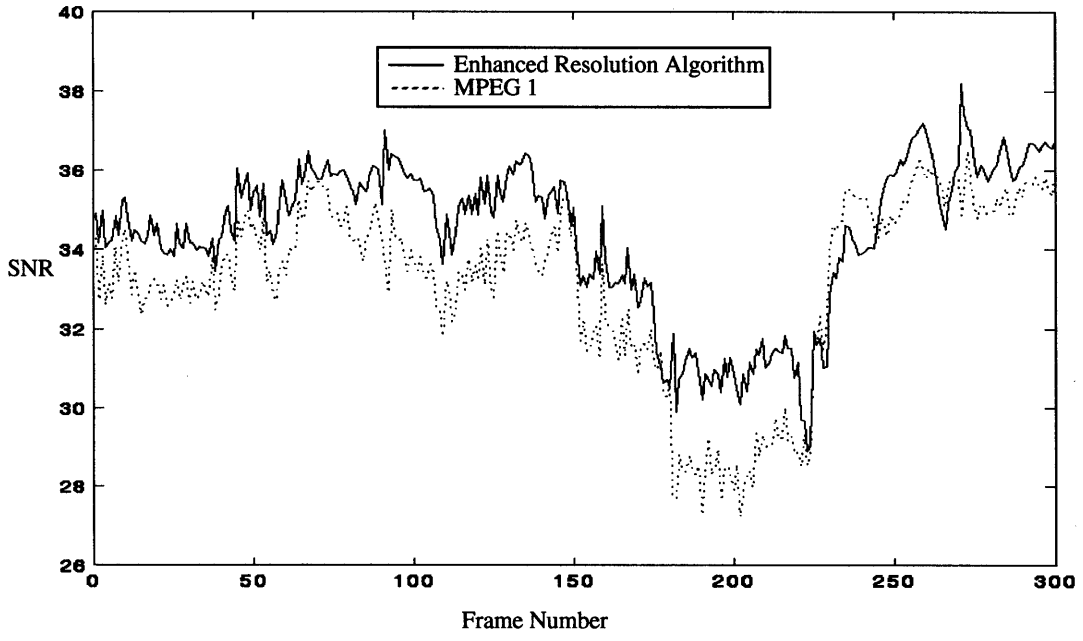


Figure 7.8 SNR for the Lecturer Sequence

As might have been expected the enhanced resolution algorithm did not perform as well for the carousel sequence. However, it still performed better than MPEG, primarily due to the more accurate motion representation.

Over the three hundred frame sequence the average SNR for the MPEG 1 coder was 29.37 dB while the enhanced resolution encoder achieved an SNR of 30.15 dB, an improvement of 0.78 dB over MPEG 1.

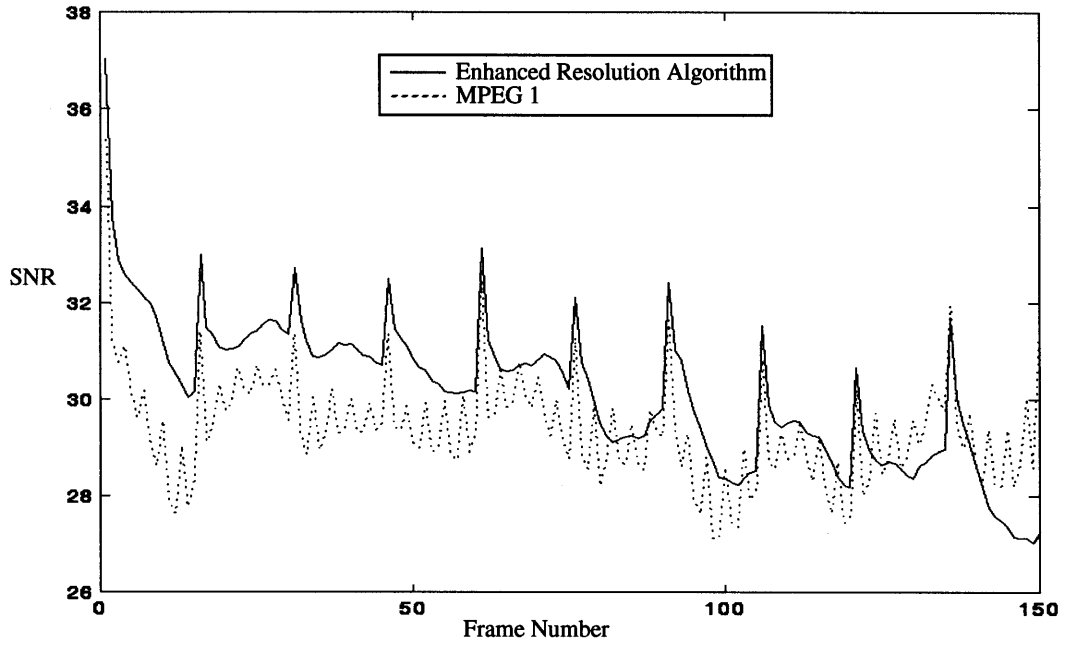


Figure 7.9 SNR for the Carousel Sequence

Chapter 8

Conclusions

8.1 Performance of Enhanced Resolution Codec

The primary aim of developing a new coding algorithm capable of including some notion of content into the encoded representation of video was achieved. Furthermore, this aim was achieved with little or no loss in picture quality when compared to MPEG.

The new representation arose from splitting the image data into background and foreground data, which were then transmitted independently from one another. This made several things possible;

- a synthetic super resolution image of the background in the scene could now be constructed, thus enabling the viewer to concentrate on items of interest in the background, e.g. writing on a whiteboard during a lecture.
- foreground elements could be removed from this background image, thus enabling the viewer to see background areas of interest that became temporarily occluded by foreground objects, e.g. writing on a whiteboard temporarily obscured by the lecturers arm.
- the resolution of the decoded image could now be determined by the decoder to some extent independently from the encoder without the need for enhancement layers transmitted in parallel to the primary bitstream.

Interestingly all of these benefits were achieved at little or no expense in terms of compression efficiency for the test sequences (lecture and carousel) when compared with the more traditional MPEG1 hybrid transform coder. In fact, even when the input to the enhanced resolution algorithm was chosen to break the assumed background model, the enhanced resolution algorithm still out performed MPEG by achieving a higher average SNR for the carousel sequence encoded at 1.2Mbit/s. This feature is quite important as it means that the enhanced resolution algorithm degrades gracefully back to a more traditional hybrid transform coder when the assumptions made during its design are broken. Thus, the enhanced resolution encoder can be used for any sequence without the risk of excessively high bit rates when compared to the MPEG algorithm.

8.2 Possible Extensions for Further Study

Although the enhanced resolution algorithm represents a significant improvement over MPEG a lot of work still remains to be done in order for it to achieve the more ambitious goals that motivated its inception. The primary aim of the thesis was to develop a new and more useful representation for encoded moving images: a representation that included content, one that allowed the viewer to manipulate images with greater freedom and understanding. Currently video is still very much a linear medium; searching for specific items of interest requires the use of wasteful techniques which result in at least one person investing a significant amount of time viewing an entire sequence. This should not be required.

Some areas for improvement in the enhanced resolution algorithm that come to mind are;

- Exploring better segmentation techniques that segment more cleanly than the motion based technique used. Perhaps the explicit inclusion of color and texture will allow more accurate segmentation to be achieved.
- More efficient and meaningful representations for foreground objects. In terms of efficiency, triangular patches appear to be the obvious choice. They may even lead to methods that will enable 3D models to be constructed for the foreground object based on the relative motion of their vertices.
- Random access and model update techniques. Currently the encoder makes an assumption about how many models should be used in the construction of the background. At the decoder the models from the previous Group of Pictures (GOP) are discarded when a new one starts, thus forfeiting the gains made by their presence. A smarter decoder with more memory than that assumed by the encoder would keep the best models from previous GOP's for inclusion with the current models.

Appendix A: Spatially Adaptive Hierarchial Block Matching

Motion estimation is somewhat of a black art, no motion estimation technique today generates a perfect output that represents the true motion of the image for all possible cases. The algorithm described here represents one attempt to improve a standard block matching motion estimator and to improve its performance both in terms of increased speed and increased accuracy. The modifications made to increase accuracy have been kept simple in keeping with the idea that the motion estimator should be as efficient as possible.

A.1 Decreasing Computational Complexity

The first and most straightforward modification made to the motion estimator was to make it hierarchical. This modification decreases the computational complexity of motion estimation considerably. A pyramid decomposition was performed on both the current image and the predictor where each level was scaled by a factor of two, both horizontally and vertically, from those above and below it. Motion estimation was then performed in a top down fashion with successive refinements being made to the motion estimates at each level until the bottoms of both pyramids were reached.

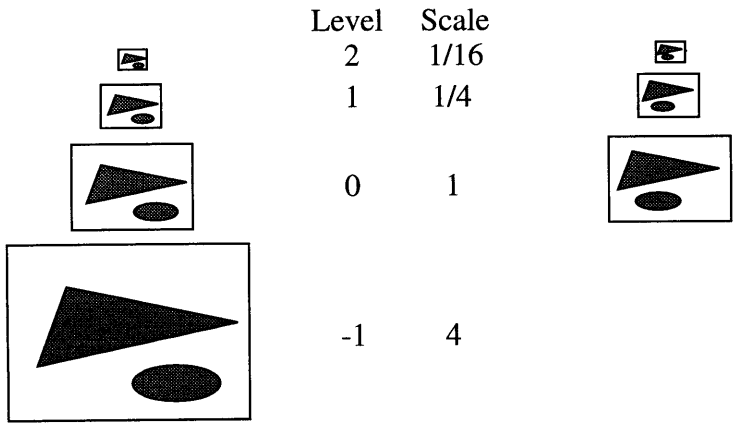


Figure A.1. Pyramid Decomposition

The refinement process between two levels consisted of two steps;

- The motion vectors from the level above were doubled to reflect the change in scale.
- Next a localized search was performed centered on the doubled motion vectors from step 1.

The computational savings over a standard non-hierarchical algorithm arise out of the fact that large motions in lower levels become much smaller in higher levels. Having computed an estimate at a highest level negates the need to perform anything but a +/- 1 pel search at lower levels as the small motion estimates computed at the highest level will eventually be scaled to their true value. The search has basically become a two dimensional bilinear one.

Having reduced the computational complexity considerably, it became necessary to examine methods to improve the motion estimator's accuracy.

A.2 Increasing Accuracy

Block matching algorithms based upon the SAD criterion are known to give unreliable estimates when the block being predicted consists of highly uniform samples. In these cases a small perturbation makes little difference to the error, so any vector is just as good as any other. Therefore, the SAD criterion will attempt to correlate, not the block data but the small amounts of noise present in both the block and the predictor, thereby generating an inaccurate prediction of the true motion.

One solution to this problem is to use the normalized correlation coefficient [5] as the error measure shown in Eqn (A.1) instead of the SAD criterion.

$$N(\delta_x, \delta_y) = \frac{E(q_1 q) + E(q_1) E(q_2)}{\sigma(q_1)\sigma(q)}$$

q_1 = orignal block

q_2 = predictor block offset by (δ_x, δ_y) **(A.1)**

However, the Normalized Correlation requires the computation of the variance of each possible matching block q_2 for each original block q_1, a requirement that increases the amount of computation required considerably. Consequently, the normalized correlation coefficient was not used.

Three methods were chosen to improve the accuracy of the motion estimates:

First, motion estimation was performed between original to original images not original to reconstructed as is common with MPEG coders. Using the reconstructed image instead of the original image makes sense for MPEG coders as accumulated errors in the reconstructed image may result in the true motion giving a larger residual than that for the true motion vector. However, as the aim of the motion estimator in the enhanced resolution codec is to find the true motion for the block, and not to minimize the error residual, original to original motion estimation was used.

Secondly, instead of searching the candidate vector space using the usual raster scan method, a spiral search method was used which tended to favor smaller motion vectors rather than those oriented to the top left as with the raster search. In an effort to reduce the number of outliers in the motion field, this bias towards smaller motion vectors is further increased for highly uniform blocks by multiplying the final SAD coefficient by a factor dependent on the blocks variance. The spiral search order is shown in Figure A.2,

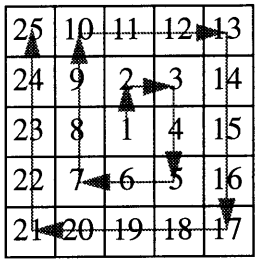


Figure A.2 Spiral Search Method

The third and final method employed to increase the accuracy of the motion estimates was to perform median filtering of the individual components of each vector using a 3 by 3 separable median filter after each refinement stage in the hierarchy. This had the effect of eliminating outliers that were missed by the spiral search modifications. Unfortunately, median filtering also removed the smallest valid regions. However, as the overhead associated with transmitting the regions could be quite expensive this was deemed to be only a minor flaw.

Appendix B: Affine Model Estimation

The process of affine model estimation consists of fitting a model to a set of data points. Not all data points will fit the model therefore, the process of estimating the model involves several iterations of model estimation and subsequent pruning of the data points that do not fit the model. The model being used here is the standard 6 parameter model below

$$\begin{aligned} v_x &= a_x + b_x x + c_x y \\ v_y &= a_y + b_y x + c_y y \end{aligned} \tag{B.1}$$

The parameters a_x, b_x, c_x, a_y, b_y and c_y are computed from the set of motion vector estimates (v_x,v_y) generated by the motion estimator. As the motion vectors consist of two components this process actually becomes one of fitting two models to two sets of data points with the only dependency between the two being that the data points used to construct each model be the same. The least squares method for fitting a single plane to either v_x or v_y is given below

$$\begin{aligned} v &= a + bx + cy \\ \text{LSE} &= \sum_R (v - (a + bx + cy))^2 \\ &= \sum_R v^2 - 2v(a + bx + cy) + (a + bx + cy)^2 \end{aligned} \tag{B.2}$$

Differentiating the error term by a, b & c and then setting them to zero yields three linear equations in terms of the three unknown parameters,

$$\begin{aligned} \frac{d}{da}LSE \rightarrow a\sum_R 1 + b\sum_R x + c\sum_R y - \sum_R v(x,y) &= 0 \\ \frac{d}{db}LSE \rightarrow a\sum_R x + b\sum_R x^2 + c\sum_R xy - \sum_R xv(x,y) &= 0 \\ \frac{d}{dc}LSE \rightarrow a\sum_R y + b\sum_R xy + c\sum_R y^2 - \sum_R yv(x,y) &= 0 \end{aligned} \tag{B.3}$$

These Equations can be written in matrix form,

$$\begin{bmatrix} \sum_R 1 & \sum_R x & \sum_R y \\ \sum_R x & \sum_R x^2 & \sum_R xy \\ \sum_R y & \sum_R xy & \sum_R y^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_R v(x,y) \\ \sum_R xv(x,y) \\ \sum_R yv(x,y) \end{bmatrix} \quad \text{(B.4)}$$

Solving for a, b & c is then simply achieved via diagonalization and back substitution.

It should be noted that the choice of which data points to include in the region R is critical in the determination of the affine parameters. If outliers are included in the solution then they will tend to skew the parameters away from their correct values. In order to prevent this happening, the region R is recursively computed by estimating the parameters for the current set of data points and then redefining the region R to include only those points which are "close" to the model. The amount of deviation allowed is reduced with each step until the affine model closely reflects the data points in the region R which contains few, if any, outliers. The block diagram for the algorithm used in this thesis is presented below in Figure B.1.

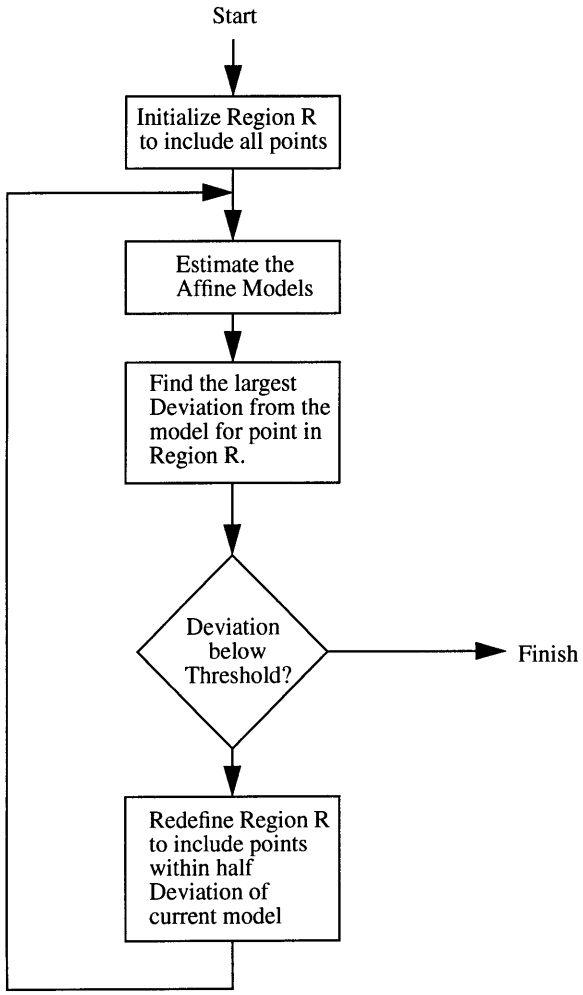


Figure B.1. Block Diagram of 6 Parameter Affine Estimation Algorithm

Appendix C: Enhanced Resolution Codec Bit Stream Definition

The presentation style of the enhanced resolution codec bit stream syntax has been deliberately modeled on the style used by the MPEG2 Committee Draft. The reason for this lies in the fact that the MPEG2 syntax was used as a starting point for the Enhanced Resolution syntax. As many syntactical elements are identical between the two encoders those elements which are unique to the Enhanced Resolution Encoder are printed in *italics,* all syntactical elements printed in normal script should be considered to have identical meaning to their equivalents in the MPEG2 standard.

C.1 Video Bitstream Syntax

C.1.1 Video Sequence

video_sequence() {	No. of bits	Mnemonic
next_start_code()		
sequence_header()		
do {		
do {		
if (nextbits() == group_start_code) {		
group_of_pictures_header()		
}		
picture_header()		
picture_data()		
} while ((next_bits() == picture_start_code) ||		
(next_bits() == group_start_code))		
if (next_bits() != sequence_end_code) {		
sequence_header()		
}		
} while (nextbits() != sequence_end_code)		
sequence_end_code	32	

C.1.2 Sequence Header

sequence_header() {	No of bits	Mnemonic
sequence_header_code	32	bslbf
horizontal_size_value	12	uimsbf
vertical_size_value	12	uimsbf
aspect_ratio_information	4	uimsbf
frame_rate_code	4	uimsbf
bit_rate_value	18	uimsbf
marker_bit	1	bslbf
vbv_buffer_size	10	uimsbf
load_intra_quantizer_matrix	1	
if (load_intra_quantizer_matrix)		
intra_quantizer_matrix[64]	64*8	uimsbf
load_non_intra_quantizer_matrix	1	
if (load_non_intra_quantizer_matrix)		
non_intra_quantizer_matrix[64]	64*8	uimsbf
}		

C.1.3 Group of Pictures Header

group_of_pictures_header() {	No of bits	Mnemonic
group_start_code	32	bslbf
time_code	25	
}		

C.1.4 Picture Header

picture_header() {	No of bits	Mnemonic
picture_start_code	32	bslbf
temporal_reference	10	uimsbf
picture_coding_type	3	uimsbf
vbv_delay	16	uimsbf
if ((picture_coding_type == BG_MODEL)||		
(picture_coding_type == MODEL_DELTA)) {		
bg_lock	1	
if (bg_lock)		
bg_lock_model_num	4	uimsbf
}		
}		

C.1.5 Picture Data

picture_data() {	No of bits	Mnemonic
if (picture_coding_type == MODEL_DELTA) {		
background_motion()		
num_foreground_objectsA	7	uimsbf
fg_seg_block_size	5	
while (fg_object_to_be_transmitted) {		
foreground_object()		
}		
}		
error_residual_data()		
}		

C.1.6 Background Affine Model

background_motion() {	No of bits	Mnemonic
bg_motion_start_code	32	bslbf
a_scale	4	uimsbf
a_size	4	uimsbf
if (a_size != 0) {		
ax_sign	1	
ax	1 - 16	vlclbf
ay_sign	1	
ay	1 - 16	vlclbf
}		
bc_scale	5	uimsbf
bx_cy_size	5	uimsbf
if (b_size != 0) {		
bx_sign	1	
bx	1 - 32	vlclbf
cy_sign	1	
cy	1 - 32	vlclbf
}		
by_cx_size	5	uimsbf
if (by_cx_size != 0 {		
by_sign	1	
by	1 - 32	vlclbf
cx_sign	1	
cx	1 - 32	vlclbf
}		
}		

The 6 parameters; ax, bx, cx, ay, by, and cy, represent real numbers that have been encoded as integers. To reconstruct the true floating point value of a parameter the following scaling must be performed.

$$truevalue = sign(signbit)\frac{value\&\ ((1 \ll size) - 1)}{1 \ll scale} \quad \text{(B.5)}$$

where *value* is the decoded integer of length *size* in bits.

- The scales for the a, and b & c parameters are encoded separately.
- If size is 0 then the parameters to which the size value refers shall be set to 0.
- The signbit for each parameter is set to "1" to indicate a negative parameter, conversely "0" indicates that the parameter is positive.

C.1.7 Foreground Object

foreground_object {	No of bits	Mnemonic
foreground_object_start_code	32	bslbf
object_num	7	uimsbf
horizontal_f_code	4	uimsbf
vertical_f_code	4	uimsbf
first_block_px	7	uimsbf
first_block_py	7	uimsbf
do {		
if (not_first_block)		
relative_block_position()		
if (picture_coding_type == MODEL_DELTA) {		
mv_ref	1	
motion_vector()		
}		
} while (next_bits != start_code)		
}		

The mv_ref parameter indicates the reference for the following motion vector as follows;

- "1" means that the motion vector is referenced to the previous valid motion vector EXCEPT in the case where pos_delta_type == 0x01 in which case the motion vector corresponding to the 1st block transmitted in the previous row shall be used as the reference vector.
- "0" means that the motion vector is referenced to the motion vector of the block immediately above in the previous row.

C.1.8 Relative Block Position

relative_block_position() {	No of bits	Mnemonic
pos_delta_type	1 - 3	vlclbf
if ((pos_delta_type = fg_delta) ||		
(pos_delta == bg_delta)) {		
while (nextbits() == macroblock_escape)		
macroblock_escape	11	vlclbf
macroblock_addres_increment	1 - 11	vlclbf
} else {		
while (nextbits() == negative_delta_escape)		
negative_delta_escape	9	vlclbf
while (nextbits() == positive_delta_escape)		
positive_delta_escape	9	vlclbf
pos_delta_code	1 - 11	vlclbf
}		
}		

C.1.9 Motion Vector

motion_vector() {	No of bits	Mnemonic
motion_horizontal_code	1 - 11	vlcbf
if ((horizontal_f != 1) &&		
(motion_horizontal_code != 0))		
motion_horizontal_r	1 - 8	uimsbf
motion_vertical_code	1 - 11	vlclbf
if ((vertical_f != 1) &&		
motion_vertical_code != 0))		
motion_vertical_r	1 - 8	uimsbf
}		

C.1.10 Error Residual Data

error_residual_data() {	No of bits	Mnemonic
error_residual_start_code	32	bslbf
do {		
error_slice()		
} while (nextbits() == error_slice_start_code)		
}		

C.1.11 Error Slice

error_slice() {	No of bits	Mnemonic
error_slice_start_code	32	bslbf
quantizer_sclae_code	5	uimsbf
do {		
residual_macroblock()		
} while (nextbits != start_code)		
next_start_code()		
}		

C.1.12 Residual Macroblock

residual_macroblock() {	No of bits	Mnemonic
while (next_bits() == macroblock_escape)		
macroblock_escape	11	vlclbf
macroblock_address_increment	1 - 11	vlclbf
residual_macroblock_type	1 - 3	vlclbf
if (macroblock_quant)		
quantizer_scale_code	5	uimsbf
if (macroblock_pattern)		
coded_block_pattern_420	3 - 9	vlclbf
for (i=0; i<block_count; i++) {		
block(i)		
}		
}		

C.1.13 Block

block(i) {	No of bits	Mnemonic
if (pattern_code[i]) {		
if (macroblock_intra) {		
if (i<4) {		
dct_dc_size_luminance	2 - 9	vlclbf
if (dct_dc_size_luminance != 0)		
dct_dc_differential	1 - 11	uimsbf
} else {		
dct_dc_size_chrominance	2 - 10	vlclbf
if (dct_dc_size_chrominance != 0)		
dct_dc_size_differential	1 - 11	uimsbf
}		
} else {		
first_dct_coefficient	...	
}		
while (nextbits() != End of block)		
Subsequent DCT coefficients	...	
End of block	...	
}		
}		

C.2 Variable Length Code Tables

The variable length codes described here have been derived by either reducing or making additions to similar tables present in the MPEG standard.

- Table 1, pos_delta_type, is the exception to this rule and was derived by inspection.
- Table 2, residual_macroblock_type for MODEL frames was derived from macroblock_type table for I frames by extracting the applicable modes and truncating unnecessarily long VLC codes through the removal of leading zeros.
- Table 3, residual_macroblock_type for MODEL_DELTA frames was derived from the macroblock_type table for P frames by extracting the applicable modes and truncating unnecessarily long VLC codes through the removal of leading zeroes.
- Table 4, pos_delta_code, was derived from the motion_code vlc table by adding two escape codes to allow deltas in excess of +/- 16. The use of the escape codes is identical to that used in MPEG2 for case where the macroblock_increment is greater than 33.

TABLE C.1. Variable length codes for pos_delta_type

pos_delta_type VLC code	Description
1	fg_block_skip
01	bg_block_skip, relative to start of previous row
001	bg_block_skip

TABLE C.2. Variable length codes for residual_macroblock_type in model frames

residual_macroblock_type VLC Code	Description
1	Intra
01	Intra, Quant

TABLE C.3. Variable length codes for residual_macroblock_type in model delta frames

residual_macroblock_type VLC Code	Description
1	Inter
011	Intra
001	Inter, Quant
0001	Intra, Quant

TABLE C.4. Variable length codes for pos_delta_code

VLC Code	pos_delta_code
0000 0001 1	negative_delta_escape
0000 0011 001	-16
0000 0011 011	-15
0000 0011 101	-14
0000 0011 111	-13
0000 0100 001	-12
0000 0100 011	-11
0000 0100 11	-10
0000 0101 01	-9
0000 0101 11	-8
0000 0111	-7
0000 1001	-6
0000 1011	-5
0000 111	-4
0001 1	-3
0011	-2
011	-1
1	0
010	1
0010	2
0001 0	3
0000 110	4
0000 1010	5
0000 1000	6
0000 0110	7
0000 0101 10	8
0000 0101 00	9
0000 0100 10	10
0000 0100 010	11
0000 0100 000	12
0000 0011 110	13
0000 0011 100	14
0000 0011 010	15
0000 0011 000	16
0000 0001 0	positive_delta_escape

References

[1] ISO/IEC DIS 10918-1, ITU- T Rec.T.81 (JPEG) "Information Technology - Digital compression and coding of continuous-tone still images", 1992

[2] ISO/IEC 11172 (1993) "Information technology - Coding of moving picture and associated audio for digital storage media as up to about 1.5 Mbit/s"

[3] ISO/IEC CD 13818-2, Recommendation H.262, "Generic Coding of Moving Pictures and Associated Audio, Committee Draft, Seoul, November 1993

[4] E.H. Adelson, "Layered Representations for Image Coding", MIT Media Lab, Vision & Modelling Group Technical Report No 181, Dec 1992.

[5] D.H. Ballard and C.M. Brown, "Computer Vision", Prentice Hall 1982

[6] J.R. Bergen and R.Hinorani, "Hierarchial Motion-Based Frame Rate Conversion", April 1990.

[7] V.M. Bove Jr., "Entropy-based depth from focus", J.Opt.Soc. Am. A/Vol. 10, No. 4, April 1993

[8] H. Brusewitz, "Motion compensation with triangles", in Proc. 3rd International Conf. on 64kbit Coding of Moving Video, free session, Sept. 1990

[9] R. Clarke, *Transform Coding of Images*, Academic Press, 1985

[10] J. Foley, A. van Dam, S. Feiner & J. Hughes, *Computer Graphics*, Addison Wesley 1990.

[11] E.W. Forgy, "Cluster analysis of multivariate data: efficiency vs. interpretability of classifications", abstract, Biometrics, Vol. 21, 1965.

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

[13] H. Holtzman, "Three-Dimensional Representations of Video using Knowledge Based Estimation", Master's Thesis, MIT Media Lab, 1991.

[14] B.K.P. Horn, *Robot Vision*, The MIT Press/Mc Raw Hill, 1986.

[15] D.A. Huffman, "A Method for the Construction of Minimum Redundancy Codes", Proceedings IRE, 1962, Vol. 40, pp. 1098-1101.

[16] M. Irani and S. Peleg, "Image Sequence Enhancement Using Multiple Motions Analysis", Dept. of Computer Science, The Hebrew University of Jerusalem, Israel, Technical Report 91-15, December 1991.

[17] D.H. Kelly, "Visual responses to time-dependent stimuli", J.Opt. Soc. Am., 51, pp. 422-429, 1961.

[18] P. McLean, "Structured Video Coding", Master's Thesis, MIT Media Lab,1992.

[19] J.S. Lim, *Two-Dimensional Signal and Image Processing*, Prentice Hall, 1993.

[20] A.B. Lippman and R.G. Kermode, "Generalized Predictive Coding of Movies", Picture Coding Symposium (PCS '93), Lausanne, Switzerland, March 1993.

[21] H.G Musmann, M. Hotter and J. Ostermann, "Object-Oriented Analysis-Synthesis Coding of Moving Images", Signal Processing: Image Communication, Vol. 1, No. 2, October 1989.

[22] Y. Nakaya and H. Harashima, "Motion Compensation at Very low Rates", Picture Coding Symposium (PCS '93), Lausanne, Switzerland, March 1993.

[23] A.N. Netravali and B.G. Haskell, *Digital Pictures*, Pienum Press, New York, 1988.

[24] S. Okubo, "Requirements for high quality video coding standards", Signal Process., Vol.4, No.2, April 1992.

[25] W.F. Schreiber, *Fundamentals of Electronic Imaging Systems*, Springer-Verlag, Heidelberg 1986

[26] J.B. Stampleman, "Scalable Video Compression", Master's Thesis, MIT Media Lab 1992.

[27] G.J. Sullivan, "Multiple-Hypothesis Motion Compensation for Low Bit-Rate Video Coding, IEEE ICASSP '93Sept 1993.

[28] L. Teodosio, "Salient Stills", Master's Thesis, MIT Media Lab 1992.

[29] G.K. Wallace, "The JPEG Compression Standard", Communications of the ACM, April 1991, Vol. 34, No. 4, pp. 30 - 44.

[30] J.Y.A. Wang, E.H. Adelson, and U. Desai, "Applying Mid-level Vision Techniques for Video and Data Compression and Manipulation", Proceedings of the SPIE: Digital Video Compression on Personal Computers: Algorithms and Technologies, vol. 2187, San Jose, February 1994

Acknowledgments

So many people, so much help for so long a time.....

Andy Lippman, for periodically raking me over the coals and challenging me to think.

Edward Adelson and Michael Biggar, for being my readers and putting up with email messages and Fed Ex-ed copies to Florida.

Walter Bender and Constantine Sapuntzakis for help with the Salient Stills and affine warps.

Henry Holtzman, for answering my many many dumb questions, keeping the alpha alive and finally for being a really great guy.

Linda Peterson, for helping to de-pig garbled verbiage and purge creative punctuation.

Gillian Galloway and Jill Toffa, my spies on the second floor who kept me informed of what was really going on.

Mike Bove, Shawn Becker and John "Wad" Watlington, for countless clarifications and helpful pointers.

Eng Khor and Joe Kang for their help in writing the MPEG encoder and decoder.

Mark, for putting up with me not only as an office mate but also as room mate.

Cris, for being a true friend and a source of inspiration and understanding in times of stress.