

Digital Video Processing

MPEG2 for Digital Television

Dr. Anil Kokaram anil.kokaram@tcd.ie

This section continues with more details about MPEG2

- MPEG2 Visual
- Coding I, P, B Frames
- MPEG2 Syntax
- MPEG2 Systems
- What it looks like

Thus far this course has covered some basic aspects of image and video processing. The underlying idea has been to show how many DSP concepts extend to multidimensions and that these ideas can be found at the root of many interesting applications. Although there are many systems in which digital video processing achieves some important task (e.g. object tracking for computer vision, noise reduction for surveillance video) the main mass market systems that employ these concepts is image and video communications. Digital Television, DVD, Internet Video Broadcasting and slowly emerging Mobile Telephone as VideoPhone are all combining to make video communications systems of paramount industrial importance at the start of this century.

Figure 1 gives an idea about the evolution of topics within this course. In this handout the idea is to delve deeper into Digital Video Communications technology and in particular video compression. Despite the fact that standards documents exist that describe the major adopted standards in detail, it is not a trivial task to implement useable systems. This is because there are no standard solutions to the fundamental signal processing problems of data reconstruction, change detection and motion estimation. It is in those areas that different companies need to be able to compete in order to build better and marketable video communications systems. It is also precisely in these areas that the advanced information needed is not readily available in a formal setting. This course expects to satisfy that need.

We begin with a discussion about MPEG2 that will then lead to an understanding of the shortcomings of traditional models for video.

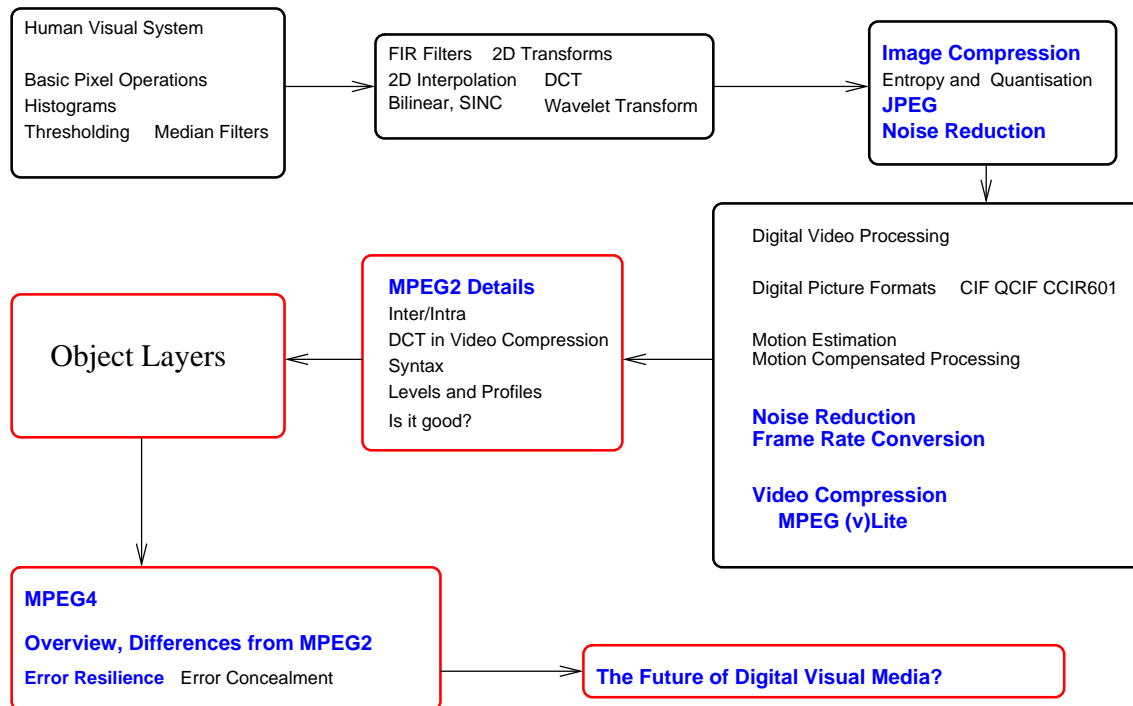


Figure 1: The progression of the course on Digital Video Processing

1 The need for compression

There has been no real justification for the need for data compression in this course. It has almost been taken for granted that the reader understands that without data compression almost all digital communications would not be possible. However, it is worth noting that there are three aspects of data manipulation that drive compression technologies. These are

- The available bandwidth of communications media

There are physical and virtual limits on communications using particular channels. Physical limits are determined by the material itself e.g. Optic Fibre, Twisted Pair Cable, Coaxial Cable, The atmosphere. Then there are various signalling overheads associated with the communications schemes that are chosen for various media. For instance, *hand-off* is a major contributor to overhead in mobile, wireless communications.

Medium	Bitrate (approx)
Telephone Modems	Up to 30 kbits/sec
ADSL Modems	Mbits/sec
ISDN	46 to 1920 kbits/sec
LAN	10 to 100 Mbits/sec
Gigabit Ethernet	Gbits/sec
Terrestrial TV	18 to 20 Mbits /sec
Cable TV	20 to 40 Mbits /sec

Improving technology does imply that with time these limits generally increase. Two major contributors to this are the design of better signal equalisation techniques and more robust modulation schemes.

- The available bandwidth of media hosts

Media servers that handle media storage and distribution have finite limitations. Their processing speed is limited and the bandwidth of the attached storage systems is limited. This means that there is always a maximum number of users that can access the server resources. It is there important to minimise the amount of data that is required to be fetched in response to each user request for media (e.g. a movie, or an audio track). There are of course alternative scenarios of distributed servers or “users-as- servers” but the limitations on maximum users can be less restrictive if efficient data storage is employed.

- The storage capacity of digital media and devices

Media/Device	Bitrate (approx)
Compaq IPaq	(64) Mbytes
CD-ROM	600 MBytes, 1.4 Mbits/sec
DVD	4 GBytes, 9 to 10 Mbits/sec
Hard Drive	72 Gbytes, 160 Mbits/sec

Each one of these bottlenecks adds additional limitations to the overall system. Consider that raw digital video, PAL Format, CCIR rec601, requires about 160 Mbits/sec (approx 20 Mbytes/sec). Raw digital audio (CD quality) requires around 670 Kbits/sec, so this adds a negligible amount. This implies that a 2.5 hour movie would need more than 170 Gigabytes of storage space (several hard drives worth) and viewing would only be possible across Optic Fibre links!

There is therefore an important need for compression to yield good quality images at factors of about 30 or more at standard (601) resolution.

2 An introduction to MPEG2

As a sequel to the JPEG standards committee, the Moving Picture Experts Group (MPEG) was set up in the mid 1980s to agree standards for video sequence compression.

Their first standard was MPEG-I, designed for CD-ROM applications at 1.5Mb/s, and their more recent standard, MPEG-II, is aimed at broadcast quality TV signals at 4 to 10 Mb/s and is also suitable for high-definition TV (HDTV) at 20 Mb/s. We shall not go into the detailed differences between these standards, but the principal difference is that MPEG-1 targeted storage of data on various Digital Storage Media (DSMs) while MPEG-2 targeted data compression for Digital Picture Transmission and in particular included a standard for the compression of digital interlaced television pictures.

In any multimedia transmission system that involves compression the following issues become important.

Compression There are at least three fundamentally different types of multimedia data sources: pictures, audio and text. Different compression techniques are needed for each data type. Each piece of data has to be identified with unique codewords for transmission.

Sequencing The compressed data from each source is scanned into a sequence of bits. This sequence is then packetised for transport. The problem here is to identify each different part of the bitstream uniquely to the decoder, e.g. header information, DCT coefficient information.

Multiplexing The audio and video data (for instance) has to be decoded at the same time (or approximately the same time) to create a coherent signal at the receiver. This implies that the transmitted *elementary data streams* should be somehow combined so that they arrive at the correct time at the decoder. The challenge is therefore to allow for identifying the different parts of the multiplexed stream and to insert information about the timing of each elementary data stream.

Media The compressed and multiplexed data has to be stored on some DSM and then later (or live) broadcast to receivers across air or other links. Access to different Media channels (including DSM) is governed by different constraints and this must somehow be allowed for in the standards description.

Errors Errors in the received bitstream invariably occur. The receiver must cope with errors such that the system performance is robust to errors or it degrades in some graceful way.

Bandwidth The bandwidth available for the multimedia transmission is limited. The transmission system must ensure that the bandwidth of the bitstream does not exceed these limits. This problem is called *Rate Control* and applies both to the control of the bitrate of the *elementary data streams* and the multiplexed stream.

Multiplatform The coded bitstream may need to be decoded on many different types of device with varying processor speeds and storage resources. It would be interesting if the transmission system could

provide a bitstream which could be decoded to varying extents by different devices. Thus a low capacity device could receive a lower quality picture than a high capacity device that would receive further features and higher picture quality. This concept applied to the construction of a suitable bitstream format is called *Scalability*.

We will not discuss Rate Control for MPEG-2 in any detail. Errors caused by transmission will be illustrated but solutions will be left to the discussion about MPEG-4. The issues of **Multiplexing** and **Media** are dealt with in the *Systems* specification of the MPEG standard. The creation of the elementary data streams is the heart of the standard and this course will cover the Visual standard only.

3 Profiles and Levels

The MPEG standards have been defined to cope with a wide variety of picture formats and frame rates. However not all decoders and encoders will be able to cope with all possible combinations of input picture formats. Therefore a hierarchy of data formats was specified such that the capability of a Codec could be defined in terms of a combination of various input data options allowed by the standard. Recall that the standard does **not** define encoders or decoders themselves, it *only* defines the format of a bitstream. This format implicitly defines much of the decoder but creation of the bitstream (the encoder) is outside the standard.

Profiles in the standard define sets of algorithms to be used for compression. Generally, each Profile is a superset of algorithms found in the Profile below as shown in the following table.

Profile	Algorithms
HIGH	Supports all functionality of the SPATIAL Scalable Profile plus 3 layers with SNR and Spatial Scalable coding modes 4:2:2 YUV picture format
SPATIAL Scalable	Supports all functionality of the SNR Scalable Profile plus Spatial Scalable coding modes (2 layers) 4:0:0 YUV
SNR Scalable	Supports all functionality of the MAIN Profile plus SNR Scalable coding (2 layers) 4:2:0 YUV
MAIN (core)	Nonscalable coding supporting all functionality of the SIMPLE profile plus B-picture prediction modes
SIMPLE	Nonscalable coding supporting coding progressive and interlaced video random access I, P-picture prediction modes 4:2:0 YUV

Within each Profile there are a number of options for the range of parameters that can be supported. The upper bounds of each such range e.g. picture size, frame rate, bit rate, is defined in the scope of a *Level* as follows.

Level	Upper Bound on Parameters
HIGH	1920 samples/line , 1152 lines/frame 60 frames/sec 80 Mbits/sec
HIGH 1440	1440 samples/line, 1152 lines/frame 60 Frames/sec 60 Mbits/sec
MAIN	720 samples/line, 576 lines/frame 30 Frames/sec 30 Mbits/sec
LOW	352 samples/line, 288 lines/frame 30 frames/sec 4 Mbits/sec

Most Digital Television (DTV) MPEG-2 equipment for home reception operates in at least the MAIN Profile using the MAIN Level for parameter bounds. The MAIN Profile is also the MPEG-2 core algorithm set.

The table below gives some idea about the typical compression ratios possible with MPEG-2 applied to various picture formats.

	Video Resolution (pels \times lines \times frames)	Uncompressed Bitrate	Compressed Bitrate
PAL Video	$720 \times 576 \times 25Hz$	≈ 160 Mbits/sec	4 to 9 Mbits/sec
HDTV Video	$1920 \times 1080 \times 60Hz$	≈ 1400 Mbits/sec	18 to 30 Mbits/sec
QCIF Videophone	$176 \times 144 \times 25Hz$	≈ 10 Mbits/sec	10 to 30 Kbits/sec

As of 20001, a new standard is being planned for Digital Cinema (perhaps $1920 \times 1080 \times 24$) by the Digital Cinema working group of MPEG.

4 MAIN Profile Overview

As stated earlier, video compression in MPEG2 is achieved by a combination of Transform coding and DPCM. In general, most frames are coded by subtracting a motion compensated version of a pervious frame and then taking the DCT of the resulting DFD (Displaced Frame Difference). There are 3 types of coded frames, *Intra-Coded* (I) frames, *Predictively Coded* (P) frames and *Bi-directionally coded* (B) frames. *I* frames are not predicted at all and are coded using the same process

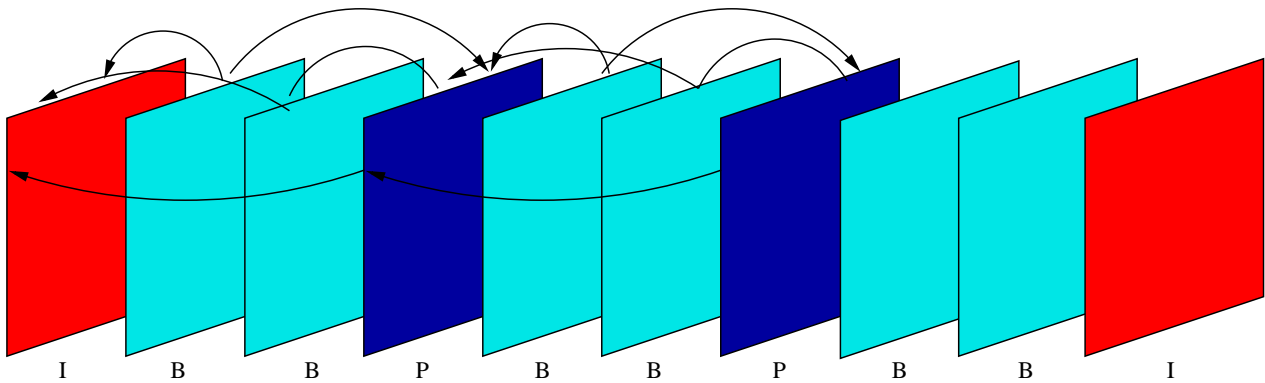


Figure 2: A typical Group of Pictures (GOP) in MPEG2

as JPEG. *P* frames are predicted from the previous I frame. B frames are created by prediction from the nearest pair of I or P frames that are next/previous in the sequence of frames. This is shown in figure 2.

I Frames are necessary because the accuracy of motion based prediction decays with time. Over increasing time intervals, objects are more likely to undergo ‘complex motion’ that violates the underlying image sequence model being used. Inserting I frames into the coded sequence allows the decoder to periodically refresh its buffers with a more accurate frame from which to start predictive coding.

4.1 Compression

The basic unit of compression in MPEG-2 is the Group of Pictures (GOP). It is a small number of consecutive frames (typically 12) that always starts with an Intra coded frame and that is followed by P and possibly B frames. The GOP can be decoded without reference to other frames outside the group. It allows actions like Fast Forward, Fast Rewind by allowing the decoder to begin decoding at any start of a GOP in the entire video sequence. It also allows editing and splicing of MPEG-2 streams.

4.1.1 On Motion Compensated Prediction

To understand how prediction can help with video compression, The top row of figure 4 shows a sequence of images of the *Suzie* sequence. It is QCIF (176×144) resolution and at a frame rate of 30 frames/sec. We will use a numerical evaluation of Entropy in bits/pel to show how much compression is achieved by Motion Compensated prediction, and then we will investigate the significance of GOP lengths as well as the importance of B Frames. **See Appendix A to recall a previous discussion on Entropy.**

We have already discussed that Transform coding of images yields significant levels of compres-

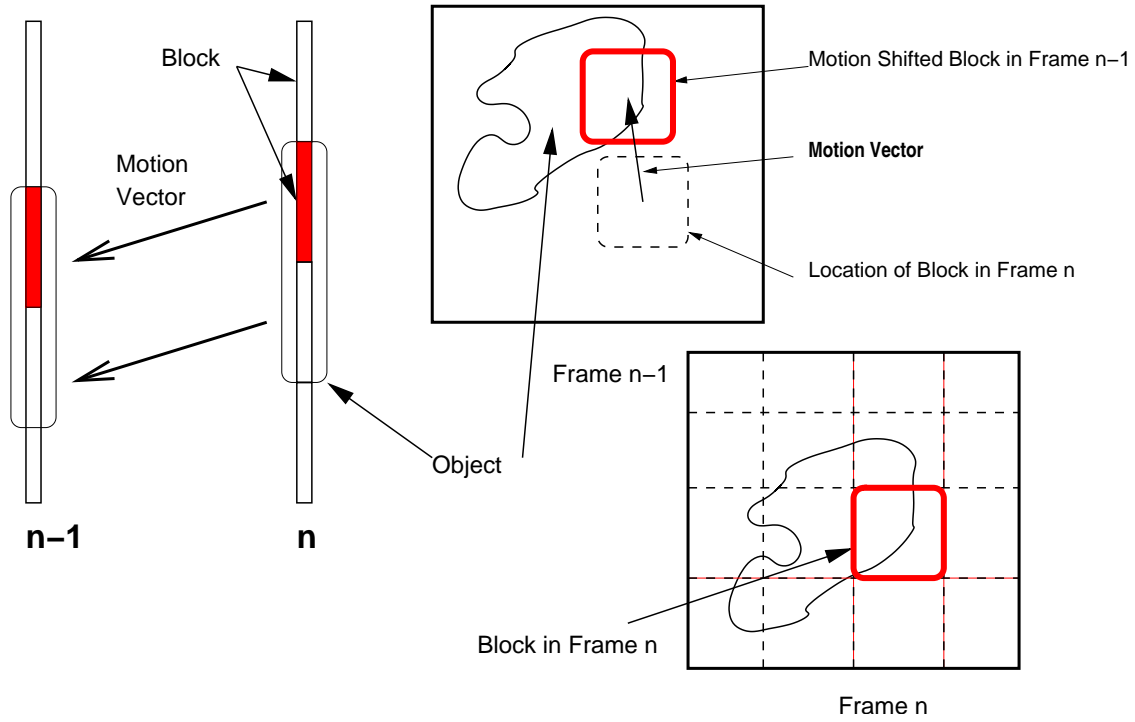


Figure 3: Explaining how motion compensation works.

sion, e.g. JPEG. Therefore a first step at compressing a sequence of data is to consider each picture separately. Consider using the 2D DCT of 8×8 blocks. Figure 5 shows in the top left hand corner, a plot of the Entropy of the DCT coefficients in the 8×8 block calculated for Frame 52. The DCT coefficients for each frame of Suzie are shown in the second row of figure 4. The use of the DCT on the raw image data yields a compression of the original 8 bits/pel data to about 0.8 bits/pel on each frame. This is shown in figure 6 which plots (+-) the Entropy in bits/pel versus frame number for a subset of 11 frames. Note that the DCT coefficients have NOT been quantised using the standard JPEG Quantisation matrix. Instead, a Q_{step} of 15 is used throughout this section.

We know that most images in a sequence are mostly the same as the frames nearby except with different object locations. Thus we can propose that the image sequence obeys a simple predictive model (discussed in previous lectures) as follows:

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x})) + e(\mathbf{x}) \quad (1)$$

where $e(\mathbf{x})$ is some small prediction error that is due to a combination of noise and “model mismatch”. Thus we can measure the prediction error at each pixel in a frame as

$$e(\mathbf{x}) = I_n(\mathbf{x}) - I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x})) \quad (2)$$

This is the motion compensated prediction error, sometimes referred to as the Displaced Frame Difference (DFD). The only model parameter required to be estimated is the motion vector $\mathbf{d}(\cdot)$. Assume for the moment that we use Hierarchical Block Matching¹ to estimate these vectors.

¹See Previous Lectures

Figure 3 illustrates how motion compensation can be applied to predict any frame from any previous frame using motion estimation. The figure shows block based motion vectors being used to match every block in frame n with the block that is most similar in frame $n - 1$. The difference between the corresponding pixels in these blocks according to equation 2 is the prediction error.

In MPEG, the situation shown in figure 3 (where frame n is predicted by a motion compensated version of frame $n - 1$) is called **Forward Prediction**. The block that is to be constructed i.e. frame n is called the **Target Block**. The frame that is supplying the prediction is called the **Reference Picture**, and the resulting data used for the motion compensation (i.e. the displaced block in frame $n - 1$) is the **Prediction Block**.

4.1.2 IPPP

The Fourth row of Figure 4 shows the prediction error of each frame of the Suzie sequence starting from the first frame as a reference. A three level Block Matcher was used with 8×8 blocks and a motion threshold for motion detection of 1.0 at the highest resolution level. The accuracy of the search was ± 0.5 pixels. Thus the sequence is *almost* the same as that which would be coded by an MPEG-2 coder employing a GOP consisting of IPPP frames in that sequence. Each DFD frame is the difference between frame n and a motion compensated frame $n - 1$, *given the original frame $n - 1$* .

DISCUSS: Why is it *not exactly* the same as the frames that would be used in an MPEG-2 encoder?

Again, we can compress this sequence of ‘transformed’ images (including the first I frame) using the DCT of blocks of 8×8 . Figure 5 shows the Entropy of a DCT block in Frame 52. Figure 6 shows on the right (o-) that the Entropy per frame is now about 0.4 bits/pel. It shows clearly that substantial compression has been achieved over attempting to compress each image separately. Of course, you will have deduced that this was going to be the case because there is *much* less information content in the DFD frames than in the original picture data.

To confirm that it is indeed *motion compensated prediction* that is contributing most of the benefit, the 3rd row of figure 4 shows the non-motion compensated frame difference (FD) $I_n(\mathbf{x}) - I_{n-1}(\mathbf{x})$ between the frames of Suzie. The Entropy per frame (after the DCT) is also plotted in figure 6 as a comparison. There is substantially more energy in these FD frames than in the DFD frames, hence the higher Entropy. However, since there is a very large area of the sequence after frame 57 which is not moving much, this difference is not as large as one might think. At this stage then we can make two observations.

1. Correlation between frames in an image sequence is highest only along motion trajectories
2. When there is little motion, there is little to be gained from motion compensation.

DISCUSS: Why is the Entropy per frame **HIGHER** (in the first few frames after

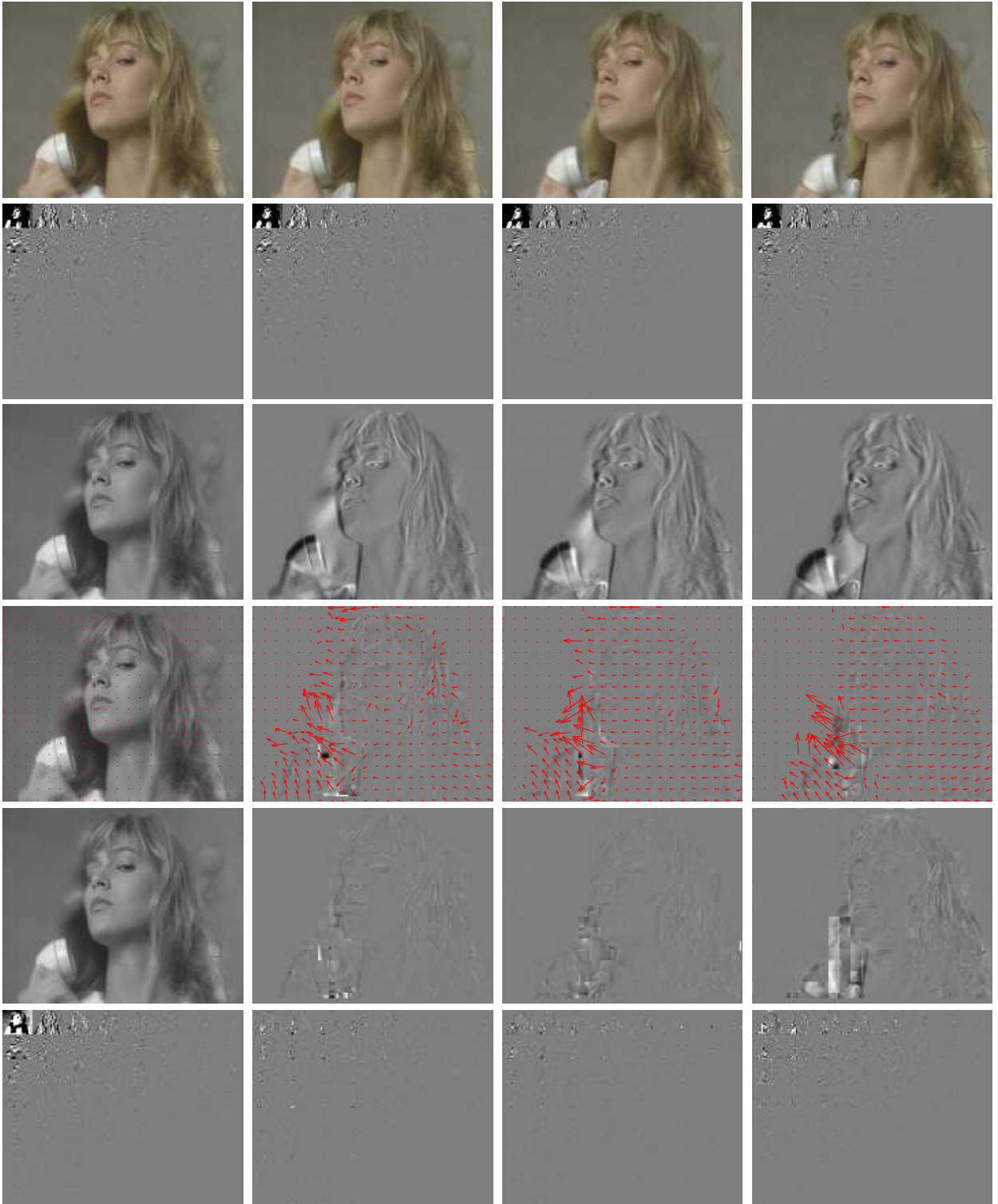


Figure 4: Frames 50-53 of the Suzie sequence processed by various means. From Top to Bottom row: Original Frames; DCT of Top Row; Non-motion compensated DFD; Motion Compensated DFD with IPPP; Motion Compensated DFD with IBBP; DCT of previous row.

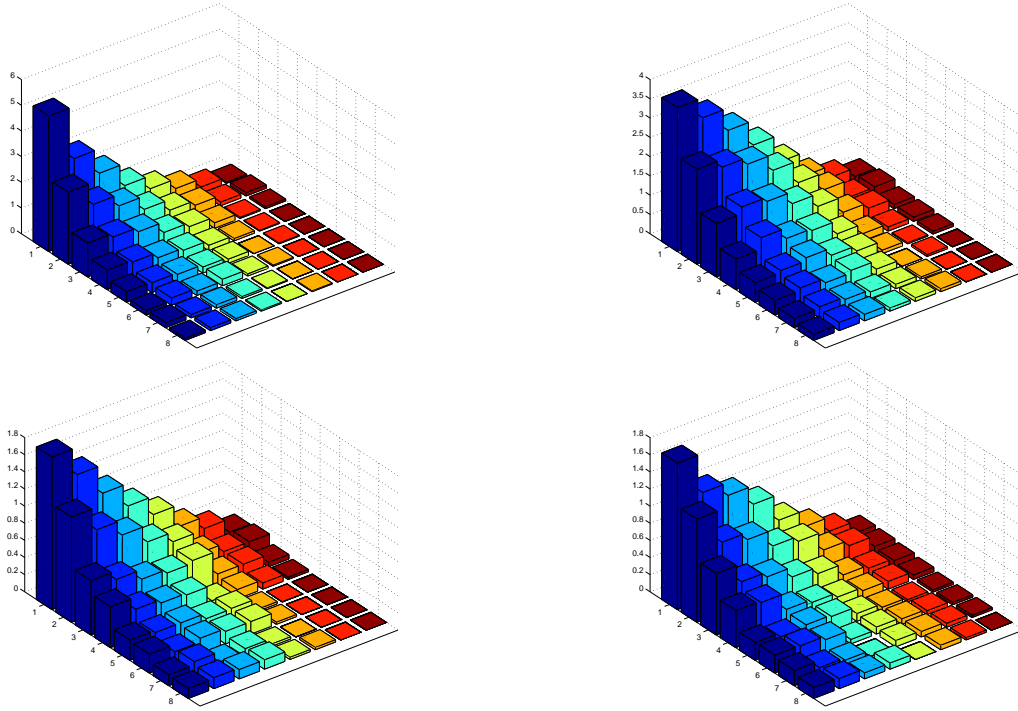


Figure 5: Entropy of DCT Coefficients of various DFDs compared for frame 52 of Suzie. Top left: DCT of I Frame, Top right: Non motion compensated DFD, Bottom left: DFD of P Frame, Bottom right: DFD of B Frame

frame 50) when using non-motion compensated dfd than using the DCT of the raw frames?

4.1.3 Those B Frames

A closer look at the DFD frame sequence in row 2 of Figure 4 shows that in frames 52 and 53 (in particular) there are some areas that show very high DFD. This is explained by observing the behaviour of Suzie in the top row. In those frames her head moves such that she uncovers or occludes some area of the background. The phone handset also uncovers a portion of her swinging hair. In the situation of uncovering, the data in some parts of frame n simply does not exist in frame $n - 1$. Thus the DFD must be high. However, the data that is uncovered in frame n , typically is also exposed in frame $n + 1$. Therefore, if we could look into the *next* frame as well as the *previous* frame we probably will be able to find a good match for any block whether it is *occluded* or *uncovered*.

Using such *Bi-directional* prediction gives much better image fidelity. A Bi-directionally predicted pixel would be defined as

$$\hat{I}_n(\mathbf{x}) = \begin{cases} I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}(\mathbf{x})) & \text{If } |DFD_b| < |DFD_f| \\ I_{n+1}(\mathbf{x} + \mathbf{d}_{n,n+1}(\mathbf{x})) & \text{If } |DFD_f| < |DFD_b| \end{cases}$$

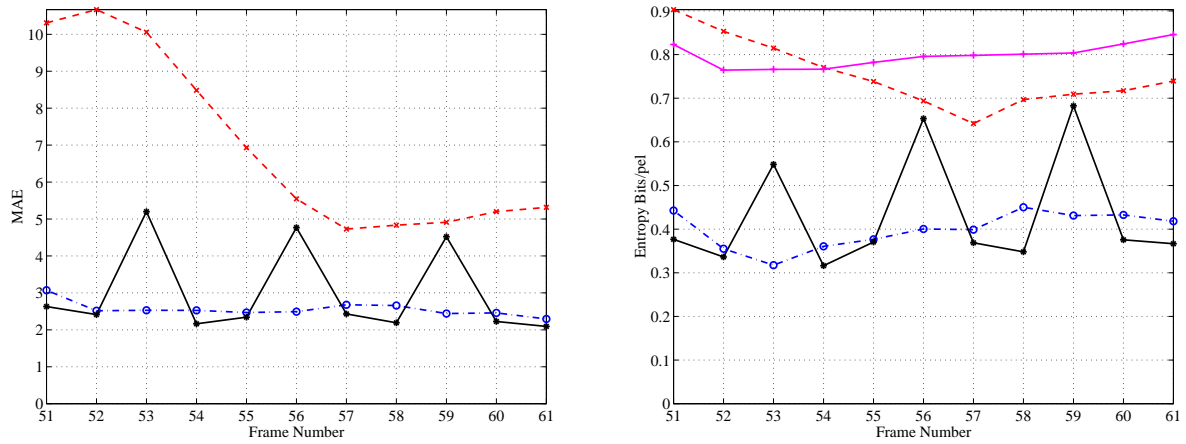


Figure 6: Comparison of MAE (left) and Entropy of the DCT of the corresponding prediction error (right) for frames 51-61 of Suzie. No motion compensation (x), IPPP (12 frame GOP) Motion Compensation (o), IBBP (12 frame GOP) Motion Compensation (*), Entropy of DCT of original frames (+) . The first I frame is not shown.

where $|DFD_b|$, $|DFD_f|$ are the DFDs employing motion compensation using the previous or next frame respectively.

In general, the detail in the Bi-Directionally predicted frames is better than in the Uni-Directionally predicted frames. The MAE plot on the left of figure 6 shows that the B frames always have a less MAE than the corresponding P frames even though the P frames are only being predicted from the previous frame. These Bi-Directionally predicted frames are the B frames used in MPEG and Figure 6 gives a final comparison of Entropies using a sequence of IBBPBBP frames. The overall entropy improvement for the B frames is not great, but it is the **visual quality of the prediction** that is of significance here.

DISCUSS: Why are there peaks in the IBBPBBP plots? And why is the Entropy for the P frames in the IBBP sequence higher than for the IPPP sequence?

4.1.4 Complications: Using the DCT in MPEG-2

Significant compression is only achieved after quantisation of the DCT coefficients. Selecting the right quantisation of the coefficients is a matter of trading off visual quality of the reconstructed pictures versus compression ratio. Typically, the coarser the quantisation, the worse the visual quality, but the higher the compression ratio. As with JPEG, the MPEG committee has selected default values for the quantisation matrices for the 8×8 DCT block.

DISCUSS: What is the difference between the Entropy of a DCT block for I and B/P frames?

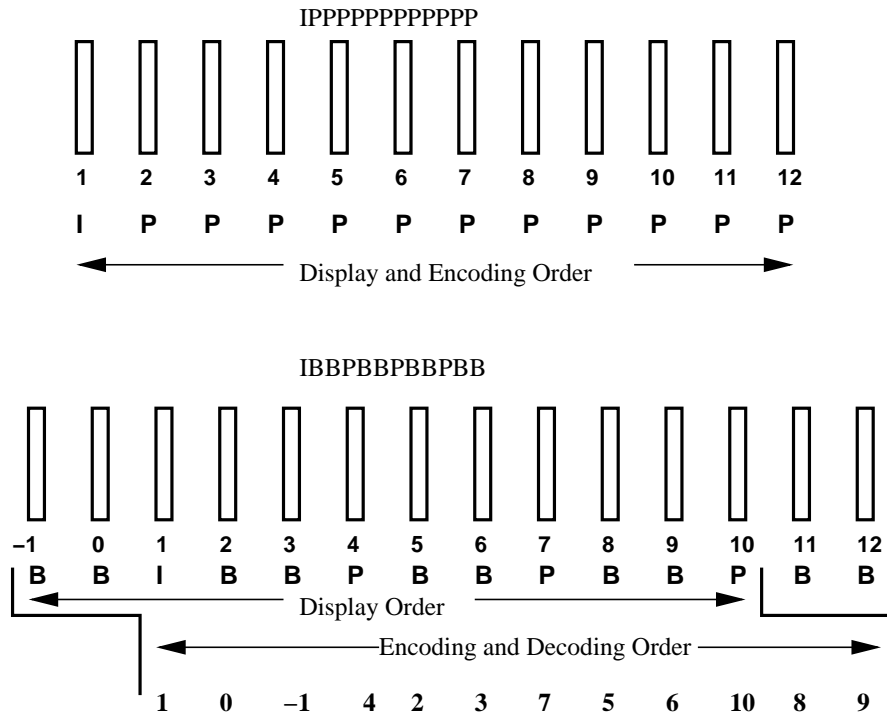


Figure 7: The order for encoding IPPPPP (top) and IBBP GOPs (bottom).

Figure 5 shows the Entropy of the DCT coefficients for Intra (raw image) frames and Inter (prediction error in predicted frames) coded images. It is clear from these plots that the Inter-coded images should employ a *different* quantisation matrix than the Intra coded images. This is precisely what MPEG have done and the two matrices are defined below.

The Default Intra Quantisation Matrix.

$$\begin{bmatrix} 8 & 16 & 19 & 22 & 26 & 27 & 29 & 34 \\ 16 & 16 & 22 & 24 & 47 & 49 & 34 & 37 \\ 19 & 22 & 26 & 27 & 29 & 34 & 34 & 38 \\ 22 & 22 & 26 & 27 & 29 & 34 & 37 & 40 \\ 22 & 26 & 27 & 29 & 32 & 35 & 40 & 48 \\ 26 & 27 & 29 & 32 & 35 & 40 & 48 & 58 \\ 26 & 27 & 29 & 34 & 38 & 46 & 56 & 69 \\ 27 & 29 & 35 & 38 & 46 & 56 & 69 & 83 \end{bmatrix}$$

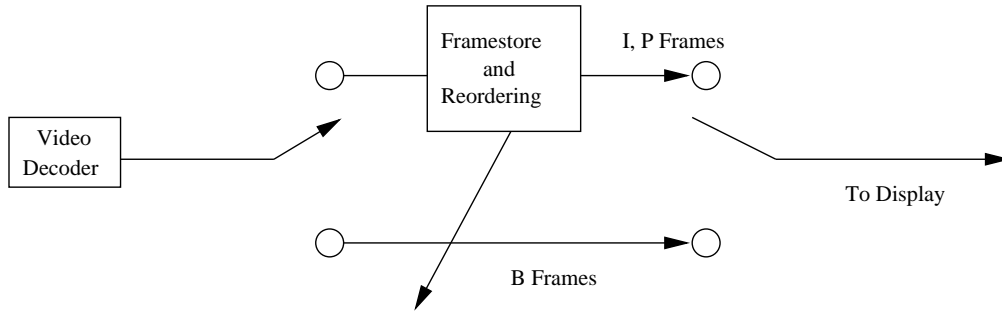


Figure 8: The reordering system at the decoder.

The Default Inter Quantisation Matrix.

$$\begin{bmatrix} 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \\ 16 & 16 & 16 & 16 & 16 & 16 & 16 & 16 \end{bmatrix}$$

4.1.5 Complications: The Frame Sequence

We have illustrated that it is useful to encode B frames in the compressed sequence. However, this implies that there must be a frame ahead as well as a frame behind in the decoder buffer before the decoding of a B frame can start. This means that the order of encoding the sequence must be different from the sequential processing of frames. Figure 7 illustrates this problem. There must be a buffer of at least 3 frames before a B frame can be encoded. Similarly, at the decoder, there must be a buffer for reordering/delaying the I and P frames so that they are displayed after the B frames arrive if necessary. Figure 8 illustrates this idea.

4.1.6 Complications: Motion

In order to reconstruct the P and B frames, motion information has to be sent. However, there is an overhead in sending motion vectors with each block. Therefore in MPEG-2 motion estimation and compensation is carried out on blocks of 16×16 and NOT 8×8 . Thus only ONE vector is transmitted per 16×16 macroblock for P Pictures and up to TWO for B Pictures.

Assuming that motion in a local area is unusually roughly the same, then it is sensible to code the motion information Differentially. Thus a Prediction Motion Vector is subtracted from the vector

for the current macroblock and the resulting difference is coded. The vector used for prediction is the vector used in the previous macroblock that was decoded. In effect, a motion model is being used that is as follows.

$$\mathbf{d}(\mathbf{x}) = \mathbf{d}(\mathbf{x} - [16, 16]) + \mathbf{v}(\mathbf{x}) \quad (3)$$

The prediction error $\mathbf{v}(\cdot)$ is the quantity that is coded.

The accuracy of the motion information transmitted is also an issue. There is a requirement to keep overhead low by quantising the vector accuracy. In MPEG-2 motion vectors are quantised to ± 0.5 pixel accuracy.

Interlaced video causes substantial difficulty for motion estimation. This is because each field is actually recorded at a different time. MPEG-2 has 5 modes for dealing with prediction of interlaced frames. The simplest mode is called Field Prediction for Field Pictures, In this mode Target Macroblocks are created by taking data from the **same** field. A second mode called Field Prediction for Frame Pictures allows a 16×16 macroblock to be rearranged into 8 top field lines at the top of the block and the remaining 8 bottom field lines at the bottom of the block. In this latter mode, there are either up to 2 or 4 vectors per block depending on whether P or B frames are being coded.

Conceptually, after field rearrangement, motion compensation behaves as for normal (progressive scan) frame pictures.

4.2 Syntax: Solving the sequencing problem

The MPEG-2 bitstream is organised with a strict hierarchy, like an onion. The layers are as follows.

1. **The Sequence Layer.** This contains an entire video sequence with thousands of frames.
2. **The GOP Layer.** This delineates a unit of frames that can be decoded independantly.
3. **The Picture Layer.** This section contains all the encoded data referring to a single I, P or B frame.
4. **The Slice Layer.** This contains the data pertaining to a set of macroblocks running from left to right across the image. It is of arbitrary length in macroblocks. A slice is not allowed to extend over the right hand end of an image, and slices must not overlap.

Slices are the MPEG-2 solution to resynchronisation. All prediction registers (e.g. for prediction of DC coefficients of each DCT block) are reset to 0 at the start of a slice. The encoder chooses the length of a slice depending on the error conditions that would be encountered.

It is at the start of a Slice that the `quantiser_scale_code` is set. This is the quantiser step size to be used for the DCT coefficients in the blocks of that slice **unless** it is optionally changed inside the code for a Macroblock.

5. **The Macroblock Layer.** This contains a 16×16 block of pixels arranged in 4 8×8 blocks. It is the motion compensation unit in that it is at this layer that vectors are associated with blocks.
6. **The Block Layer.** This contains the DCT coefficients for a single block.

4.2.1 Coding Motion

In MPEG-2 the unit of motion compensation is the Macroblock. Thus motion estimation and compensation is carried out on blocks of 16×16 and NOT 8×8 .

To differentially code the motion information the Prediction Motion Vector needs to be defined. In MPEG-2 for I and P pictures, the PMV is the PMV of the previous macroblock if it has one. Otherwise it is 0. The situation is the same for B Macroblocks (which require 2 vectors), except that if the previous macroblock only has one vector, then the PMV for both the B vectors is the same. Note that forward and backward vectors must be predicted from their counterparts in the previous frame.

The actual coding of the motion prediction error is very similar to the coding of the DC coefficients in JPEG. The prediction error is first calculated as

$$\Delta = 2(\mathbf{d} - \text{PMV}) \quad (4)$$

where \mathbf{d} is the motion vector to be coded. Recall that the motion vectors are all quantised to 0.5 pel accuracy and so δ is an integer.

MPEG adopts a much smaller code by using a form of floating-point representation, where `r_size` is the base-2 exponent and `motion_residual`, `motion_code` are used to code the polarity and precise amplitude as follows:

$$|\Delta| = (\text{motioncode} - 1)2^{\text{rsize}} + \text{motionresidual} + 1 \quad (5)$$

`r_size` is a 4 bit number and `motion_code` is an integer $0 \dots 16$.

This may seem confusing, but what is happening is that `r_size` is selecting a maximum range for the motion given by $\pm 2^{\text{rsize}} \times 16$. Each step of `motion_code` then quantises the motion vector in steps of 2^{rsize} ; and finally the `motion_residual` allows to code the remainder after division by 2^{rsize} . The table below is another way of thinking about this.

Δ (Difference)	Size r_size	Code for Size	motion_residual (in binary)
-15 – 15	0	0000	-
-31 – 31	1	0001	0,1
-63 – 63	2	0010	00,01,10,11
-127 – 127	3	0011	000, ...,011,100, ...,111
\vdots	\vdots	\vdots	
-1023, ..., -512, 512, ..., 1023	8	1000	0000 0000, 0000 0001, ... 1111 1110, 1111 1111

Thus $\Delta = 25$ may be coded by **r_size**=1, **motion_code**= $25/2 = 13$, **motion_residual**=0. Whence $25 = (13 - 1)2^1 + 0 + 1$ as in equation 5.

Only **motion_code** is Huffman coded in the above scheme, since, within a given Size, all the input values have sufficiently similar probabilities for there to be little gain from entropy coding the Motion Residual (hence they are coded in simple binary as listed). Each **motion_code** is followed by the appropriate number of Motion Residual (equal to Size) to define the sign and magnitude of the coefficient difference exactly.

It is the **r_size** that is similar to the JPEG Size value, and the combination of **motion_residual**, **motion_code** that is similar to the JPEG Additional Bits value.

There are only 17 Motion Codes (0-17) to be Huffman coded, so specifying the code table can be very simple and require relatively few bits in the header. A sample of the Huffman code tables for the **motion_code** is as follows ('s' is the sign bit, 0 for +ve, 1 for -ve).

Value of motion_code	VLC Code word
0	1
1	01s
2	001s
3	0001s
4	0000 11s
5	0000 101s
6	0000 100s
7	0 0000 011s
\vdots	\vdots
16	00 0000 1100s

In MPEG-2 the Size is sent at the start of each Frame (picture layer) and remains fixed for the whole frame. The parameters **motion_code** and **motion_residual** can be sent in each Macroblock for coding each vector as required.

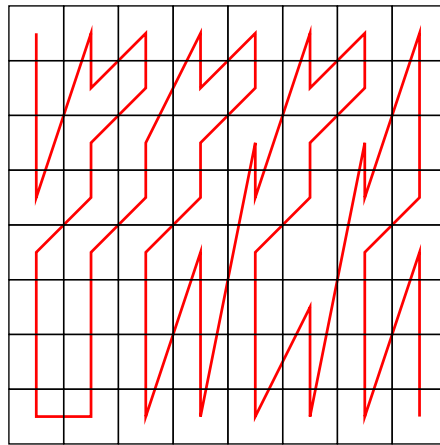


Figure 9: The alternate scan for 8×8 DCT blocks useful for Interlaced Frames

If $\Delta = 0$ it is signalled by setting `motion_code = 0`.

4.2.2 Coding the DCT coefficients

This is almost identical to JPEG, except that the run,level codes are different for Intra and Non-intra macroblocks. The range of the coefficients is ± 2047 . There are also two alternate scan patterns for the DCT blocks. The usual zig-zag scan is one option, and the other is a slightly altered scan to allow for coding of interlaced TV frames. This is because the correlation pattern for interlaced frames is obviously different. The alternate scan often gives better compression when there is significant motion. Figure 9 shows this whacky scan.

Another option that can be used in MPEG-2 is Field DCT coding. In this option, the top and bottom 8 lines in a macroblock are rearranged to contain lines from the same field. This improves the vertical correlation and hence the energy compaction in the DCT.

Chrominance Macroblocks are not reordered in the Main Profile.

4.3 MPEG2 Systems: Solving the Multiplexing and Media Problems

The main need for a SYSTEMS MPEG-2 specification is to standardise the serialisation of *all the data* associated with a multimedia stream. The following are the issues

- Extraction of the compressed data from each single source called *Elementary Streams*.
- Multiplexing the Elementary Streams together with synchronisation information.
- Defining a system reference *time* concept.

Each elementary stream (audio, video) is divided into *packets* that are interleaved by the encoder. Rate control may be achieved both at this interleaving level and inside the elementary stream generation itself. It is sometimes necessary to regulate the channel capacity allocated to elementary streams e.g. when some elementary streams require temporarily greater bandwidth. This is called *Statistical Multiplexing*.

Variation in packet lengths within the elementary streams can cause delays. These must be controlled. This is achieved by the use of Time Stamps and Clock References. There are two kinds of Time Stamps: a Decoding Time Stamp (DTS) and a Presentation Time Stamp (PTS). The DTS flags when an associated Presentation Unit is to be decoded. The PTS flags when that unit is to be passed to the output device for display. The systems specification defines a systems clock: STC. When the decoder STC advances to a DTS or PTS the required interaction with the input data buffer is instigated.

The basic unit of the elementary stream is the *Packetised Elementary Stream Packet*. This consists of a header followed by a payload. The PES header begins with an MPEG-2 Start Code. This code consists of a prefix (23 or more binary 0s followed by a binary 1) followed by a start code ID. The ID identifies the stream uniquely as well as the type (audio, video etc). After further information such as DTS etc, the Elementary Data itself follows until it is terminated by a PES End code.

The multiplexed data is called a *Non-elementary* stream. MPEG-2 defines 3 types of such stream. A Program Stream (PS) (with its own STC), a Transport Stream (TS) and a PES Stream. The TS is used for channels with non-negligible error. The Program is constructed of **Packs** of multiplexed data. Each *Pack* contains a number of PES packets from various elementary streams. The Pack header begins with a start code followed by a Systems Clock reference. These Program streams are suitable for Digital Storage media or other transmission with negligible error.

PES Packets are not the only kind of packetisation possible. MPEG-2 also defines a *fixed length* packet type for channels that are susceptible to error. A Transport Stream packet is of a fixed length of 188 bytes.

DISCUSS: How can I detect errors in an MPEG2 stream?

5 More information?

The book *Digital Video: An Introduction to MPEG-2*, Barry Haskell, Atul Purui and Arun Netravali, Chapman and Hall 1997 is an excellent overview of MPEG2. There was also a very good overview article in the Signal Processing magazine in September 1997 *MPEG Audio and Video Coding* on the MPEG2 standard.

Prof. Inald Lagendijk at the Delft University of Technology in The Netherlands has an excellent MPEG2 demo tool called VCDemo which he uses to teach an image and video coding course. It is

available free on the web and you will be using it in this course.

A Entropy

Entropy of source information was discussed in Peter Cullen's Digital Communications course. For an image \mathbf{I} , quantised to M levels, the entropy $H_{\mathbf{I}}$ is defined as:

$$H_{\mathbf{I}} = \sum_{n=0}^{M-1} p_n \log_2 \left(\frac{1}{p_n} \right) = - \sum_{n=0}^{M-1} p_n \log_2(p_n) \quad (6)$$

where p_n , $n = 0$ to $M - 1$, is the probability of the n^{th} quantiser level being used (often obtained from a histogram of the pel intensities).

$H_{\mathbf{I}}$ represents the mean number of bits per pel with which the quantised image \mathbf{I} can be represented using an ideal variable-length entropy code. A Huffman code usually approximates this bit-rate quite closely.

To obtain the number of bits to code an image (or subimage) \mathbf{I} containing N pels:

- A histogram of \mathbf{x} is measured using M bins corresponding to the M quantiser levels.
- The M histogram counts are each divided by N to give the probabilities p_n , which are then converted into entropies $h_n = -p_n \log_2(p_n)$. This conversion law is illustrated in fig 10 and shows that probabilities close to zero or one produce low entropy and intermediate values produce entropies near 0.5.
- The entropies h_n of the separate quantiser levels are summed to give the total entropy $H_{\mathbf{I}}$ for the subimage.
- Multiplying $H_{\mathbf{I}}$ by N gives the estimated total number of bits needed to code \mathbf{I} , assuming an ideal entropy code is available which is matched to the histogram of \mathbf{I} .

Fig 10 shows the probabilities p_n and entropies h_n for the original Lenna image assuming a uniform quantiser with a step-size $Q_{step} = 15$. The original Lenna image contained pel values from 3 to 238 and a mean level of 120 was subtracted from each pel value before the image was analysed or transformed in order that all samples would be approximately evenly distributed about zero (a natural feature of highpass subimages).

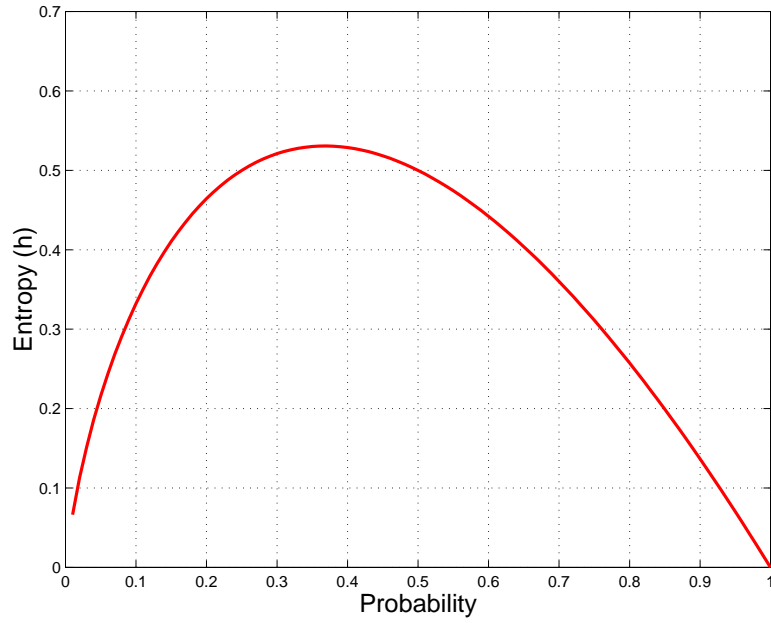


Figure 10: Conversion from probability p_n to entropy $h_n = -p_n \log_2(p_n)$.

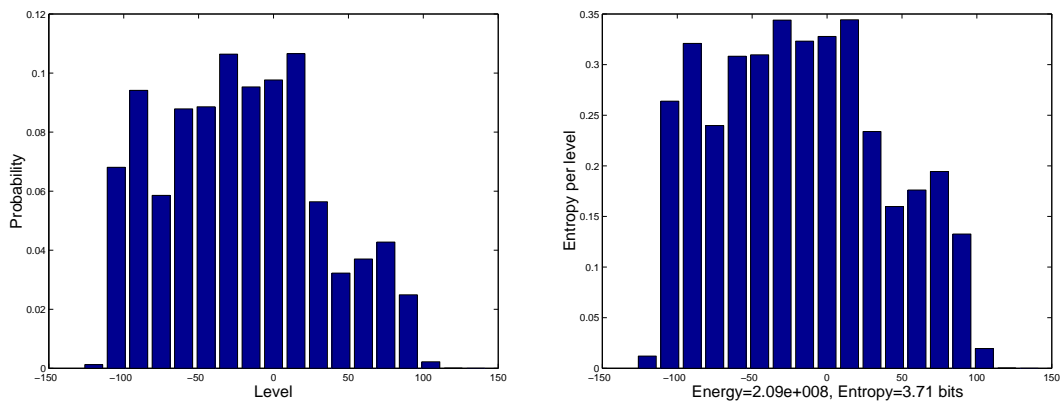


Figure 11: Probability histogram (left) and entropies (right) of the Lenna image