10

Rate, Distortion and Complexity

10.1 INTRODUCTION

The choice of video coding algorithm and encoding parameters affect the coded bit rate and the quality of the decoded video sequence (as well as the computational complexity of the video CODEC). The precise relationship between coding parameters, bit rate and visual quality varies depending on the characteristics of the video sequence (e.g. 'noisy' input vs. 'clean' input; high detail vs. low detail; complex motion vs. simple motion). At the same time, practical limits determined by the processor and the transmission environment put constraints on the bit rate and image quality that may be achieved. It is important to control the video encoding process in order to maximise compression performance (i.e. high compression and/or good image quality) whilst remaining within the practical constraints of transmission and processing.

Rate-distortion optimisation attempts to maximise image quality subject to transmission bit rate constraints. The best optimisation performance comes at the expense of impractically high computation. Practical algorithms for the control of bit rate can be judged according to how closely they approach optimum performance. Many alternative rate control algorithms exist; sophisticated algorithms can achieve excellent rate-distortion performance, usually at a cost of increased computational complexity. The careful selection and implementation of a rate control algorithm can make a big difference to video CODEC performance.

Recent trends in software-only CODECs and video coding in power-limited environments (e.g. mobile computing) mean that computational complexity is an important factor in video CODEC performance. In many application scenarios, video quality is constrained by available computational resources as well as or instead of available bit rate. Recent developments in variable-complexity algorithms (VCAs) for video coding enable the developer to manage computational complexity and trade processing resources for image quality. This leads to situations in which rate, complexity and distortion are interdependent. New algorithms are required to jointly control bit rate and computational complexity whilst minimising distortion.

In this chapter we examine the factors that influence rate-distortion performance in a video CODEC and discuss how these factors can be exploited to efficiently control coded bit rate. We describe a number of popular algorithms for rate control. We discuss the relationship between computation, rate and distortion and show how new VCAs are beginning to influence the design of video CODECs.
10.2 BIT RATE AND DISTORTION

10.2.1 The Importance of Rate Control

A practical video CODEC operates within an environment that places certain constraints on its operation. One of the most important constraints is the rate at which the video encoder is 'allowed' to produce encoded data. A source of video data usually supplies video data at a constant bit rate (a constant number of bits per second) and a video encoder processes this high, constant-rate source to produce a compressed stream of bits at a reduced bit rate. The amount of compression (and hence the compressed bit rate) depends on a number of factors. These may include:

1. The encoding algorithm (intra-frame or inter-frame, forward or bidirectional prediction, integer or sub-pixel motion compensation, DCT or wavelet, etc.).
2. The type of video material (material containing lots of spatial detail and/or rapid movement generally produces more bits than material containing little detail and/or motion).
3. Encoding parameters and decisions (quantiser step size, picture or macroblock mode selection, motion vector search area, the number of intra-pictures, etc.).

Some examples of bit rate 'profiles' are given below. Figure 10.1 plots the number of bits in each frame for a video sequence encoded using Motion JPEG. Each frame is coded independently ('intra-coded') and the bit rate for each frame does not change significantly. Small variations in bit rate are due to changes in the spatial content of the frames in the 10-frame sequence. Figure 10.2 shows the bit rate variation for the same sequence coded

![Figure 10.1 Bit-rate profile: Motion JPEG](image-url)
with H.263. The first frame is an intra-frame and following frames are P-pictures. The compression efficiency for a P-picture is approximately $10 \times$ higher than for an I-picture in this example and there is a small variation between P-pictures due to changes in detail and in movement. Coding the same sequence using MPEG-2 gives the bit rate profile shown in Figure 10.3. In this example, the initial I-picture is followed by the following sequence of picture types: B–B–P–B–B–P–B–B–I. There is clearly a large variation between the three picture types, with B-pictures giving the best compression performance. There is also a smaller variation between coded pictures of the same type (I, P or B) due to changes in detail and motion as before.
These examples show that the choice of algorithm and the content of the video sequence affect the bit rate (and also the visual quality) of the coded sequence. At the same time, the operating environment places important constraints on bit rate. These may include:

1. The mean bit rate that may be transmitted or stored.
2. The maximum bit rate that may be transmitted or stored.
3. The maximum variation in bit rate.
4. The requirement to avoid underflow or overflow of storage buffers within the system.
5. A requirement to minimise latency (delay).

Examples:

**DVD-video** The mean bit rate is determined by the duration of the video material. For example, if a 3-hour movie is to be stored on a single 4.7 Gbyte DVD, then the mean bit rate (for the whole movie) must not exceed around 3.5 Mbps. The maximum bit rate is determined by the maximum transfer rate from the DVD and the throughput of the video decoder. Bit-rate variation (subject to these constraints) and latency are not such important issues.

**Video conferencing over ISDN** The ISDN channel operates at a constant bit rate (e.g. 128 kbps). The encoded bit rate must match this channel rate exactly, i.e. no variation is allowed. The output of the video encoder is constant bit rate (CBR) coded video.

**Video conferencing over a packet-switched network** The situation here is more complicated. The available mean and maximum bit rate may vary, depending on the network routing and on the volume of other traffic. In some situations, latency and bit rate may be linked, i.e. a higher data rate may cause increased congestion and delay in the network. The video encoder can generate CBR or variable bit rate (VBR) coded video, but the mean and peak data rate may depend on the capacity of the network connection.

Each of these application examples has different requirements in terms of the rate of encoded video data. Rate control, the process of matching the encoder output to rate constraints, is a necessary component of the majority of practical video coding applications. The rate control 'problem' is defined below in Section 10.2.3. There are many different approaches to solving this problem and in a given situation, the choice of rate control method can significantly influence video quality at the decoder. Poor rate control may cause a number of problems such as low visual quality, fluctuations in visual quality and dropped frames leading to 'jerky' video.

In the next section we will examine the relationship between coding parameters, bit rate and visual quality.
10.2.2 Rate-Distortion Performance

A lossless compression encoder produces a reduction in data rate with no loss of fidelity of the original data. A lossy encoder, on the other hand, reduces data rate at the expense of a loss of quality. As discussed previously, significantly higher compression of image and video data can be achieved using lossy methods than with lossless methods. The output of a lossy video CODEC is a sequence of images that are of a lower quality than the original images.

The rate-distortion performance of a video CODEC provides a measure of the image quality produced at a range of coded bit rates. For a given compressed bit rate, measure the distortion of the decoded sequence (relative to the original sequence). Repeat this for a range of compressed bit rates to obtain the rate-distortion curve such as the example shown in Figure 10.4. Each point on this graph is generated by encoding a video sequence using an MPEG-4 encoder with a different quantiser step size $Q$. Smaller values of $Q$ produce a higher encoded bit rate and lower distortion; larger values of $Q$ produce lower bit rates at the expense of higher distortion. In this figure, ‘image distortion’ is measured by peak signal to noise ratio (PSNR), described in Chapter 2. PSNR is a logarithmic measure, and a high value of PSNR indicates low distortion. The video sequence is a relatively static, ‘head-and-shoulders’ sequence (‘Claire’). The shape of the rate-distortion curve is very typical: better image quality (as measured by PSNR) occurs at higher bit rates, and the quality drops sharply once the bit rate is below a certain threshold.

The rate-distortion performance of a video CODEC may be affected by many factors, including the following.

**Video material**

Under identical encoding conditions, the rate-distortion performance may vary considerably depending on the video material that is encoded. Figure 10.5 compares the rate-distortion.
performance of two sequences, ‘Claire’ and ‘Foreman’, under identical encoding conditions (MPEG-4, fixed quantiser step size varying from 4 to 24). The ‘Foreman’ sequence contains a lot of movement and detail and is therefore more ‘difficult’ to compress than ‘Claire’. At the same value of quantiser, ‘Foreman’ tends to have a much higher encoded bit rate and a higher distortion (lower PSNR) than ‘Claire’. The shape of the rate-distortion curve is similar but the rate and distortion values are very different.

**Encoding parameters**

In a DCT-based CODEC, a number of encoding parameters (in addition to quantiser step size) affect the encoded bit rate. An efficient motion estimation algorithm produces a small residual frame after motion compensation and hence a low coded bit rate; intra-coded macroblocks usually require more bits than inter-coded macroblocks; sub-pixel motion compensation produces a lower bit rate than integer-pixel compensation; and so on. Less obvious effects include, for example, the intervals at which the quantiser step size is varied during encoding. Each time the quantiser step size changes, the new value (or the change) must be signalled to the decoder and this takes more bits (and hence increases the coded bit rate).

**Encoding algorithms**

Figures 10.1–10.3 illustrate how the coded bit rate changes depending on the compression algorithm. In each of these figures, the decoded image quality is roughly the same but there is a big difference in compressed bit rate.

**Rate control algorithms**

A rate control algorithm chooses encoding parameters (such as those listed above) in order to try and achieve a ‘target’ bit rate. For a given bit rate, the choice of rate control
algorithm can have a significant effect on rate-distortion performance, as discussed later in this chapter.

So far we have discussed only spatial distortion (the variation in quality of individual frames in the decoded video sequence). It is also important to consider temporal distortion, i.e. the situation where complete frames are ‘dropped’ from the original sequence in order to achieve acceptable performance. The curves shown in Figure 10.5 were generated for video sequences encoded at 30 frames per second. It would be possible to obtain lower spatial distortion by reducing the frame rate to 15 frames per second (dropping every second frame), at the expense of an increase in temporal distortion (because the frame rate has been reduced). The effect of this type of temporal distortion is apparent as ‘jerky’ video. This is usually just noticeable around 15–20 frames per second and very noticeable below 10 frames per second.

10.2.3 The Rate-Distortion Problem

The trade-off between coded bit rate and image distortion is an example of the general rate-distortion problem in communications engineering. In a lossy communication system, the challenge is to achieve a target data rate with minimal distortion of the transmitted signal (in this case, an image or sequence of images). This problem may be described as follows: ‘Minimize distortion \( D \) whilst maintaining a bit rate \( R \) that does not exceed a maximum bit rate \( R_{\text{max}} \), or

\[
\min\{D\} \text{ s.t. } R \leq R_{\text{max}} \tag{10.1}
\]

(where s.t. means ‘subject to’).

The conditions of Equation 10.1 can be met by selecting the optimum encoding parameters to give the ‘best’ image quality (i.e. the lowest distortion) without exceeding the target bit rate. This process can be viewed as follows:

1. Encode a video sequence with a particular set of encoding parameters (quantiser step size, macroblock mode selection, etc.) and measure the coded bit rate and decoded image quality (or distortion). This gives a particular combination of rate \( R \) and distortion \( D \), an \( R-D \) operating point.
2. Repeat the encoding process with a different set of encoding parameters to obtain another \( R-D \) operating point.
3. Repeat for further combinations of encoding parameters. (Note that the set of possible combinations of parameters is very large.)

Figure 10.6 shows a typical set of operating points plotted on a graph. Each point represents the mean bit rate and distortion achieved for a particular set of encoding parameters. (Note that distortion \( [D] \) increases as rate \( [R] \) decreases). Figure 10.6 indicates that there are ‘bad’ and ‘good’ rate-distortion points. In this example, the operating points that give the best rate-distortion performance (i.e. the lowest distortion for a given rate \( R \)) lie close to the dotted curve. Rate-distortion theory tells us that this curve is convex (a convex hull). For a given
target rate $R_{\text{max}}$, the minimum distortion $D$ occurs at a point on this convex curve. The aim of rate-distortion optimisation is to find a set of coding parameters that achieves an operating point as close as possible to this optimum curve.

One way to find the position of the hull and hence achieve this optimal performance is by using Lagrangian optimisation. Equation 10.1 is difficult to minimise directly and a popular method is to express it in a slightly different way as follows:

$$
\min \{ J = D + \lambda R \}
$$  \hspace{1cm} (10.2)

$J$ is a new function that contains $D$ and $R$ (as before) as well as a Lagrange multiplier, $\lambda$. $J$ is the equation of a straight line $D + \lambda R$, where $\lambda$ gives the slope of the line. There is a solution to Equation 10.2 for every possible multiplier $\lambda$, and each solution is a straight line that makes a tangent to the convex hull described earlier. The procedure may be summarised as follows:

1. Encode the sequence many times, each time with a different set of coding parameters.
2. Measure the coded bit rate ($R$) and distortion ($D$) of each coded sequence. These measurements are the ‘operating points’ ($R, D$).
3. For each value of $\lambda$, find the operating point ($R, D$) that gives the smallest value $J$, where $J = D + \lambda R$. This gives one point on the convex hull.
4. Repeat step (3) for a range of $\lambda$ to find the ‘shape’ of the convex hull.

This procedure is illustrated in Figure 10.7. The ($R, D$) operating points are plotted as before. Three values of $\lambda$ are shown: $\lambda_1$, $\lambda_2$, and $\lambda_3$. In each case, the solution to $J = D + \lambda R$ is a straight line with slope $\lambda$. The operating point ($R, D$) that gives the smallest $J$ is shown in black, and these points occur on the lower boundary (the convex hull) of all the operating points.

The Lagrangian method will find the set (or sets) of encoding parameters that give the best performance and these parameters may then be applied to the video encoder to achieve optimum rate-distortion performance. However, this is usually a prohibitively complex
process. Encoding decisions (such as quantiser step size, macroblock type, etc.) may change for every macroblock in the coded sequence and so there are an extremely large number of combinations of encoding parameters.

**Example**

Macroblock 0 in a picture is encoded using MPEG-4 (simple profile) with a quantiser step size $Q_0$ in the range $2^{-31}$. The choice of $Q_1$ for macroblock 1 is constrained to $Q_0 +/- 2$. There are 30 possible values of $Q_0$; (almost) $30 \times 5 = 150$ possible combinations of $Q_0$ and $Q_1$; (almost) $30 \times 5 \times 5 = 750$ combinations of $Q_0$, $Q_1$ and $Q_2$; and so on.

The computation required to evaluate all possible choices of encoding decision becomes prohibitive even for a short video sequence. Furthermore, no two video sequences produce the same rate-distortion performance for the same encoding parameters and so this process needs to be carried out each time a sequence is to be encoded.

There have been a number of attempts to simplify the Lagrangian optimisation method in order to make it more practically useful. For example, certain assumptions may be made about good and bad choices of encoding parameters in order to limit the exponential growth of complexity described above. The computational complexity of some of these methods is still much higher than the computation required for the encoding process itself: however, this complexity may be justified in some applications, such as (for example) encoding a feature film to obtain optimum rate-distortion performance for storage on a DVD.

An alternative approach is to estimate the optimum operating points using a model of the rate-distortion characteristic. Lagrange-based optimisation is first carried out on some representative video sequences in order to find the ‘true’ optimal parameters for these sequences. The authors propose a simple model of the relationship between encoding mode selection and $\lambda$ and the encoding mode decisions required to achieve minimal distortion for a given rate constraint $R_{\text{max}}$ can be estimated from this model. The authors report a clear performance gain over previous methods with minimal computational complexity.
attempt has been made to define an optimum partition between the coded bits representing motion vector information and the coded bits representing displaced frame difference (DFD) in an inter-frame CODEC.

10.2.4 Practical Rate Control Methods

Bit-rate control in a real-time video CODEC requires a relatively low-complexity algorithm. The choice of rate control algorithm can have a significant effect on video quality and many alternative algorithms have been developed. The choice of rate control algorithm is not straightforward because a number of factors are involved, including:

- the computational complexity of the algorithm
- whether the rate control 'model' is appropriate to the type of video material to be encoded (e.g. 'static' video-conferencing scenes or fast-action movies)
- the constraints of the transmission channel (e.g. low-delay real-time communications or offline storage).

A selection of algorithms is summarised here.

*Output buffer feedback*

One of the simplest rate control mechanisms is shown in Figure 10.8. A frame of video \( i \) is encoded to produce \( b_i \) bits. Because of the variation in content of a video sequence, \( b_i \) is likely to vary from frame to frame, i.e. the encoder output bit rate is variable, \( R_v \). In Figure 10.8 we assume that the channel rate is constant, \( R_c \) (this is the case for many practical channels). In order to match the variable rate \( R_v \) to the constant channel rate \( R_c \), the encoded bits are placed in a buffer, filled at rate \( R_v \) and emptied at rate \( R_c \).

Figure 10.9 shows how the buffer contents vary during encoding of a typical video sequence. As each frame is encoded, the buffer fills at a variable rate and after encoding of each frame, a fixed number of bits \( b_c \) are removed from the buffer. With no constraint on the

![Figure 10.8 Buffer feedback rate control](image-url)
variable rate $R_v$, it is possible for the buffer contents to rise to a point at which the buffer overflows ($B_{\text{max}}$ in the figure). The black line shows the unconstrained case: the buffer overflows in frames 5 and 6. To avoid this happening, a feedback constraint is required, where the buffer occupancy $B$ is ‘fed back’ to control the quantiser step size $Q$. As $B$ increases, $Q$ also increases which has the effect of increasing compression and reducing the number of bits per frame $b_i$. The grey line in Figure 10.9 shows that with feedback, the buffer contents are never allowed to rise above about 50% of $B_{\text{max}}$.

This method is simple and straightforward but has several disadvantages. A sudden increase in activity in the video scene may cause $B$ to increase too rapidly to be effectively controlled by the quantiser $Q$, so that the buffer overflows, and in this case the only course of action is to skip frames, resulting in a variable frame rate. As Figure 10.9 shows, $B$ increases towards the end of each encoded frame and this means that $Q$ also tends to increase towards the end of the frame. This can lead to an effect whereby the top of each frame is encoded with a relatively high quality whereas the foot of the frame is highly quantised and has an obvious drop in quality, as shown in Figure 10.10. The basic buffer-feedback method tends to produce decoded video with obvious quality variations.

**MPEG-2 Test Model 5**

Version 5 of the MPEG-2 video Test Model (a reference design for MPEG-2 encoding and decoding) describes a rate control algorithm for CBR encoding that takes account of the different properties of the three coded picture types (I, P and B-pictures). The algorithm consists of three steps: bit allocation, rate control and modulation.
1. Bit allocation:
   (a) assign a target number of bits to the current GOP (based on the target constant bit rate);
   (b) assign a target number of bits $T$ to the current picture based on:
       − the 'complexity' of the previous picture of the same type (I, P, B) (i.e. the level of temporal and/or spatial activity);
       − the target number of bits for the GOP.

2. Rate control:
   (a) during encoding of the current picture, maintain a count of the number of coded bits so far, $d$;
   (b) compare $d$ with the target total number of bits $T$ and choose the quantiser step size $Q$ to try and meet the target $T$.

3. Modulation:
   (a) measure the variance of the luminance data in the current macroblock;
   (b) if the variance is higher than average (i.e. there is a high level of detail in the current region of the picture), increase $Q$ (and hence increase compression).

The aim of this rate control algorithm is to:

- achieve a target number of coded bits for the current GOP;
- deal with I, P and B-pictures separately;
- quantise areas of high detail more 'coarsely'.

\[ \text{Figure 10.10} \quad \text{Frame showing quality variation} \]
This last aim should give improved subjective visual quality since the human eye is more sensitive to coarse quantisation (high distortion) in areas of low detail (such as a smooth region of the picture).

**H.263 Test Model**

Version 8 (and later versions) of the H.263 Test Model use a rate control algorithm that consists of frame-level rate control (determines whether to skip or code the current frame) and macroblock-level rate control (calculate the quantisation step size for each macroblock).

**Frame level control** Each encoded frame adds to the encoder output buffer contents; each transmitted frame removes bits from the output buffer. If the number of bits in the buffer exceeds a threshold $M$, skip the next frame; otherwise set a target number of bits $B$ for encoding the next frame. A higher threshold $M$ means fewer skipped frames, but a larger delay through the system.

**Macroblock level control** This is based on a model for the number of bits $B_i$ required to encode macroblock $i$ (Equation 10.3):

$$B_i = A\left(K \frac{\sigma_i^2}{Q_i^2} + C\right)$$

$A$ is the number of pixels in a macroblock, $\sigma_i$ is the standard deviation of luminance and chrominance in the residual macroblock (i.e. a measure of variation within the macroblock), $Q_i$ is the quantisation step size and $K$ and $C$ are constant model parameters. The following steps are carried out for each macroblock $i$:

1. Measure $\sigma_i$.
2. Calculate $Q_i$ based on $B$, $K$, $C$, $\sigma_i$ and a macroblock weight $\alpha_i$.
3. Encode the macroblock.
4. Update the model parameters $K$ and $C$ based on the actual number of coded bits produced for the macroblock.

The weight $\alpha_i$ controls the ‘importance’ of macroblock $i$ to the subjective appearance of the image: a low value of $\alpha_i$ means that the current macroblock is likely to be highly quantised. In the test model, these weights are selected to minimise changes in $Q_i$ at lower bit rates because each change involves sending a modified quantisation parameter $DQUANT$ which means encoding an extra 5 bits per macroblock. It is important to minimise the number of changes to $Q_i$ during encoding of a frame at low bit rates because the extra 5 bits in a macroblock may become significant; at higher bit rates, this $DQUANT$ overhead is less important and we may change $Q$ more frequently without significant penalty.

This rate control method is effective at maintaining good visual quality with a small encoder output buffer which keeps coding delay to a minimum (important for low-delay real-time communications).
Example

A 200-frame video sequence, ‘Carphone’, is encoded using H.263 with TM8 rate control. The original frame rate is 30 frames per second, QCIF resolution, and the target bit rate is 64 kbps. Figure 10.11 shows the bit-rate variation during encoding. In order to achieve 64 kbps without dropping any frames, the mean bit rate should be 2133 bits per frame, and the encoder clearly manages to maintain this bit rate (with occasional variations of about ±10%). Figure 10.12 shows the PSNR of each frame in the sequence after encoding and decoding. Towards the end of the sequence, the movement in the scene increases and it becomes ‘harder’ to code efficiently. The rate control algorithm compensates for this by increasing the quantiser step size and the PSNR drops accordingly. Out of the original 200 frames, the encoder has to drop 6 frames to avoid buffer overflow.

MPEG-4 Annex L

The MPEG-4 video standard describes an optional rate control algorithm in Annex L.1, known as the Scalable Rate Control (SRC) scheme. This algorithm is appropriate for a single video object (i.e. a rectangular VO that covers the entire frame) and a range of bit rates and spatial/temporal resolutions. The scheme described in Annex L offers rate control at the frame-level only (i.e. a single quantiser step size is chosen for a complete frame). The SRC attempts to achieve a target bit rate over a certain number of frames (a ‘segment’ of frames, usually starting with an I-picture).
The SRC scheme assumes the following model for the encoder rate $R$:

$$R = \frac{X_1 S}{Q} + \frac{X_2 S}{Q^2}$$  \hspace{1cm} (10.4)

$Q$ is the quantiser step size, $S$ is the mean absolute difference of the residual frame after motion compensation and $X_1$, $X_2$ are model parameters. $S$ provides a measure of frame complexity (easier to compute than the standard deviation $\sigma$ used in the H.263 TM8 rate control scheme because the absolute difference, SAE, is calculated during motion estimation).

Rate control consists of the following steps which are carried out after motion compensation and before encoding of each frame $i$:

1. Calculate a target bit rate $R_i$, based on the number of frames in the segment, the number of bits that are available for the remainder of the segment, the maximum acceptable buffer contents and the estimated complexity of frame $i$. (The maximum buffer size affects the latency from encoder input to decoder output. If the previous frame was complex, it is assumed that the next frame will be complex and should therefore be allocated a suitable number of bits: the algorithm attempts to balance this requirement against the limit on the total number of bits for the segment.)

2. Compute the quantiser step size $Q_i$ (to be applied to the whole frame), calculate $S$ for the complete residual frame and solve Equation 10.4 to find $Q$.

3. Encode the frame.
4. Update the model parameters $X_1, X_2$ based on the actual number of bits generated for frame $i$.

The SRC algorithm differs from H.263 TM8 in two significant ways: it aims to achieve a target bit rate across a segment of frames (rather than a sequence of arbitrary length) and it does not modulate the quantiser step size within a coded frame (this can give a more uniform visual appearance within each frame but makes it difficult to maintain a small buffer size and hence a low delay). An extension to the SRC is described in Annex L.3 of MPEG-4 which supports modulation of the quantiser step size at the macroblock level and is therefore more suitable for low-delay applications. The macroblock rate control extension (L.3) is similar to H.263 Test Model 8 rate control.

The SRC algorithm is described in some detail in the MPEG-4 standard; a further discussion of MPEG-4 rate control issues can be found elsewhere.\textsuperscript{9}

### 10.3 COMPUTATIONAL COMPLEXITY

#### 10.3.1 Computational Complexity and Video Quality

So far we have considered the trade-off between bit rate and video quality. The discussion of rate distortion in Section 10.2.3 highlighted another trade-off between computational complexity and video quality. A video coding algorithm that gives excellent rate-distortion performance (good visual quality for a given bit rate) may be impractical because it requires too much computation.

There are a number of cases where it is possible to achieve higher visual quality at the expense of increased computation. A few examples are listed below:

- **DCT block size**: better decorrelation can be achieved with a larger DCT block size, at the expense of higher complexity. The $8 \times 8$ block size is popular because it achieves reasonable performance with manageable computational complexity.

- **Motion estimation search algorithm**: full-search motion estimation (where every possible match is examined within the search area) can outperform most reduced-complexity algorithms. However, algorithms such as the ‘three step search’ which sample only a few of the possible matches are widely used because they reduce complexity at the expense of a certain loss of performance.

- **Motion estimation search area**: a good match (and hence better rate-distortion performance) is more likely if the motion estimation search area is large. However, practical video encoders limit the search area to keep computation to manageable levels.

- **Rate-distortion optimisation**: obtaining optimal (or even near-optimal) rate-distortion performance requires computationally expensive optimisation of encoding parameters, i.e. the best visual quality for a given bit rate is achieved at the expense of high complexity.

- **Choice of frame rate**: encoding and decoding computation increases with frame rate and it may be necessary to accept a low frame rate (and ‘jerky’ video) because of computational constraints.
These examples show that many aspects of video encoding and decoding are a trade-off between computation and quality. Traditionally, hardware video CODECs have been designed with a fixed level of computational performance. The architecture and the clock rate determine the maximum video processing rate. Motion search area, block size and maximum frame rate are fixed by the design and place a predetermined 'ceiling' on the rate-distortion performance of the CODEC.

Recent trends in video CODEC design, however, require a more flexible approach to these trade-offs between complexity and quality. The following scenarios illustrate this.

**Scenario 1: Software video CODEC**

Video is captured via a capture board or ‘webcam’. Encoding, decoding and display are carried out entirely in software. The ‘ceiling’ on computational complexity depends on the available processing resources. These resources are likely to vary from platform to platform (for example, depending on the specification of a PC) and may also vary depending on the number of other applications contending for resources. Figure 10.13 compares the resources available to a software CODEC in two cases: when it is the only intensive application running, the CODEC has most of the system resources available, whereas when the CODEC must contend with other applications, fewer processing cycles are available to it. The computational resources (and therefore the maximum achievable video quality) are no longer fixed.

**Scenario 2: Power-limited video CODEC**

In a mobile or hand-held computing platform, power consumption is at a premium. It is now common for a processor in a portable PC or personal digital assistant to be ‘power-aware’, e.g. a laptop PC may change the processor clock speed depending on whether it is running from a battery or from an AC supply. Power consumption increases depending on the activity of peripherals, e.g. hard disk accesses, display activity, etc. There is therefore a need to manage and limit computation in order to maximise battery life.

![Figure 10.13 Available computational resources](image)
These scenarios illustrate the need for a more flexible approach to computation in a video CODEC. In this type of scenario, computation can no longer be considered to be a 'constant'. CODEC performance is now a function of three variables: computational complexity, coded bit rate and video quality. Optimising the complexity, rate and distortion performance of a video CODEC requires flexible control of computational complexity and this has led to the development of variable complexity algorithms for video coding.

10.3.2 Variable Complexity Algorithms

A variable complexity algorithm (VCA) carries out a particular task with a controllable degree of computational overhead. As discussed above, computation is often related to image quality and/or compression efficiency: in general, better image quality and/or higher compression require a higher computational overhead.

**Input-independent VCAs**

In this class of algorithms, the computational complexity of the algorithm is independent of the input data. Examples of input-independent VCAs include:

- **Frame skipping**: encoding a frame takes a certain amount of processing resources and 'skipping' frames (i.e. not coding certain frames in the input sequence) is a crude but effective way of reducing processor utilisation. The relationship between frame rate and utilisation is not necessarily linear in an inter-frame CODEC: when the frame rate is low (because of frame skipping), there is likely to be a larger difference between successive frames and hence more data to code in the residual frame. Frame skipping may lead to a variable frame rate as the available resources change and this can be very distracting to the viewer. Frame skipping is widely used in software video CODECs.

- **Motion estimation (ME) search window**: increasing or decreasing the ME search window changes the computational overhead of motion estimation. The relationship between search window size and computational complexity depends on the search algorithm. Table 10.1 compares the overhead of different search window sizes for the popular n-step search algorithm. With no search, only the (0, 0) position is matched; with a search window of +/- 1, a total of nine positions are matched; and so on.

<table>
<thead>
<tr>
<th>Search window</th>
<th>Number of comparison steps</th>
<th>Computation (normalised)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0.03</td>
</tr>
<tr>
<td>+/- 1</td>
<td>9</td>
<td>0.27</td>
</tr>
<tr>
<td>+/- 3</td>
<td>17</td>
<td>0.51</td>
</tr>
<tr>
<td>+/- 7</td>
<td>25</td>
<td>0.76</td>
</tr>
<tr>
<td>+/- 15</td>
<td>33</td>
<td>1.0</td>
</tr>
</tbody>
</table>
Pruned DCT: a forward DCT (FDCT) processes a block of samples (typically $8 \times 8$) and produces a block of coefficients. In a typical image block, many of the coefficients are zero after quantisation and only a few non-zero coefficients remain to be coded and transmitted. These non-zero coefficients tend to occupy the lower-frequency positions in the block. A 'pruned' DCT algorithm only calculates a subset of the $8 \times 8$ DCT coefficients (usually lower frequencies), reducing the computational overhead of the DCT.\textsuperscript{10,11} Examples of possible subsets are shown in Figure 10.14: the 'full' $8 \times 8$ DCT may be reduced to a $4 \times 4$ or $2 \times 2$ DCT, producing only low-frequency coefficients. However, applying a pruned DCT to all blocks means that the small (but significant) number of high-frequency coefficients are lost and this can have a very visible impact on image quality.

Input-dependent algorithms

An input-dependent VCA controls computational complexity depending on the characteristics of the video sequence or coded data. Examples include the following.

Zero testing in IDCT In a DCT-based CODEC operating at medium or low bit rates, many blocks contain no AC coefficients after quantisation (i.e. only the DC coefficient remains, or no coefficients remain). This may be exploited to reduce the complexity of the IDCT (which must be calculated in both the encoder and the decoder in an inter-frame CODEC). Each row or column of eight coefficients is tested for zeros. If the seven highest coefficients are all zero, then the row or column will contain a uniform value (the DC coefficient) after the IDCT. In this case, the IDCT may be skipped and all samples set to the DC value:

```c
if (F1 = F2 = F3 = F4 = F5 = F6 = F7 = 0) {
    f0 = f1 = f2 = f3 = f4 = f5 = f6 = f7 = F0
} else {
    [calculate the IDCT..]
}
```
There is a small overhead associated with testing for zero; however, the computational saving can be very significant and there is no loss of quality. Further input-dependent complexity reductions can be applied to the IDCT. 12

**FDCT complexity reduction** Many blocks contain few non-zero coefficients after quantisation (particularly in inter-coded macroblocks). It is possible to predict the occurrence of some of these blocks before the FDCT is carried out so that the FDCT and quantisation steps may be skipped, saving computation. The sum of absolute differences (SAD or SAE) calculated during motion estimation can act as a useful predictor for these blocks. SAD is proportional to the energy remaining in the block after motion compensation. If SAD is low, the energy in the residual block is low and it is likely that the block will contain little or no data after FDCT and quantisation. Figure 10.15 plots the probability that a block contains no coefficients after FDCT and quantisation, against SAD. This implies that it should be possible to skip the FDCT and quantisation steps for blocks with an SAD of less than a threshold value $T$:

```plaintext
if (SAD < T) {
    set block contents to zero
} else {
    calculate the FDCT and quantize
}
```

If we set $T = 200$ then any block with SAD < 200 will not be coded. According to the figure, this ‘prediction’ of zero coefficients will be correct 90% of the time. Occasionally (10% of the time in this case), the prediction will fail, i.e. a block will be skipped that should have been encoded. The reduction in complexity due to skipping FDCT and quantisation for some blocks is therefore offset by an increase in distortion due to incorrectly skipped blocks. 13, 14

![Figure 10.15 Probability of zero block vs. SAD](image)
Input-dependent motion estimation  A description has been given\textsuperscript{15} of a motion estimation algorithm with variable computational complexity. This is based on the nearest neighbours search (NNS) algorithm (described in Chapter 6), where motion search positions are examined in a series of ‘layers’ until a minimum is detected. The NNS algorithm is extended to a VCA by adding a computational constraint on the number of layers that are examined at each iteration of the algorithm. As with the SAD prediction discussed above, this algorithm reduces computational complexity at the expense of increased coding distortion. Other computationally scalable algorithms for motion estimation algorithms are described elsewhere.\textsuperscript{16,17}

10.3.3 Complexity-Rate Control

The VCAs described above are useful for controlling the computational complexity of video encoding and decoding. Some VCAs (such as zero testing in the IDCT) have no effect on image quality; however, the more flexible and powerful VCAs (such as zero DCT prediction) do have an effect on quality. These VCAs may also change the coded bit rate: for example, if a high proportion of DCT operations are ‘skipped’, fewer coded bits will be produced and the rate will tend to drop. Conversely, the ‘target’ bit rate can affect computational complexity if VCAs are used. For example, a lower bit rate and higher quantiser scale will tend to produce fewer DCT coefficients and a higher proportion of zero blocks, reducing computational complexity.

Figure 10.16  Complexity-rate-distortion surface
It is therefore not necessarily correct to treat complexity control and rate control as separate issues. An interesting recent development is the emergence of complexity-distortion theory. Traditionally, video CODECs have been judged by their rate-distortion performance as described in Section 10.2.2. With the introduction of VCAs, it becomes necessary to examine performance in three axes: complexity, rate and distortion. The ‘operating point’ of a video CODEC is no longer restricted to a rate-distortion curve but instead lies on a rate-distortion-complexity surface, like the example shown in Figure 10.16. Each point on this surface represents a possible set of encoding parameters, leading to a particular set of values for coded bit rate, distortion and computational complexity.

Controlling rate involves moving the operating point along this surface in the rate-distortion plane; controlling complexity involves moving the operating point in the complexity-distortion plane. Because of the interrelationship between computational complexity and bit rate, it may be appropriate to control complexity and rate at the same time. This new area of complexity-rate control is at a very early stage and some preliminary results can be found elsewhere.

10.4 SUMMARY

Many practical video CODECs have to operate in a rate-constrained environment. The problem of achieving the best possible rate-distortion performance is difficult to solve and optimum performance can only be obtained at the expense of prohibitively high computational cost. Practical rate control algorithms aim to achieve good, consistent video quality within the constraints of rate, delay and complexity. Recent developments in variable complexity coding algorithms enable a further trade-off between computational complexity and distortion and are likely to become important for CODECs with limited computational resources and/or power consumption.

Bit rate is one of a number of constraints that are imposed by the transmission or storage environment. Video CODECs are designed for use in communication systems and these constraints must be taken into account. In the next chapter we examine the key ‘quality of service’ parameters required by a video CODEC and provided by transmission channels.

REFERENCES