Advance Access published on June 24, 2005 doi:10.1093/comjnl/bxh112

Complexity Estimation of the H.264 Coded Video Bitstreams

HARI KALVA AND BORKO FURHT

Department of Computer Science and Engineering, Florida Atlantic University, Boca Raton, FL 33431, USA
Email: hari, borko@cse.fau.edu

The emerging H.264 video coding standard offers a flexible coding algorithm for use in a range of applications from broadcast TV to low bitrate mobile applications. The flexibility comes with complexity. The H.264 video coding algorithm is complex and requires substantial amount of resources for encoding and decoding. The resources required to decode an H.264 video varies depending upon the encoding options used. Understanding the resources required to play the video at the receiver is especially important for mobile devices with limited amount of resources. In this paper, we present an overview of the H.264 video coding standard, complexity analysis of H.264 video, and its implications for resource management and adaptive coding for mobile devices. We examine high level bitstream characteristics and its suitability for estimating the bitstream complexity and receiver resource consumption. We show that high level metrics can be used to estimate receiver resource consumption with reasonable accuracy.

Received 3 February 2005; revised 31 March 2005

1. INTRODUCTION

The H.264 standard is intended for use in a wide range of applications including high quality and high bitrate digital video applications such as digital versatile disk (DVD) and digital TV as well as the low bitrate applications such as video delivery to mobile devices. However, the computing and communication resources of the end user terminals make it impossible to use the same encoded video content for all applications. For example, the high bitrate video used for a digital TV broadcast cannot be used for streaming video to a mobile terminal. For delivery to mobile terminals, we need video content that is encoded at lower bitrate and lower resolution suitable for low-resource mobile terminals. Pre-encoding video at a few discrete bitrates leads to inefficiencies. Furthermore, the receiver capabilities available such as central processing unit (CPU), available battery and available bandwidth may vary during a session, and a pre-encoded video stream cannot meet such dynamic needs. To make full use of the receiver capabilities and deliver video suitable for a receiver, resource-adaptive video encoding and transcoding is necessary [1].

The systems that deliver video to low resource terminals such as mobile phones should transcode or encode video to a bitrate and/or resolution that suits the receiving device. The capabilities of these mobile devices vary continuously with the available bandwidth and battery power. Even the availability of the computing resources (CPU, memory, screen resolution, and co-processor and software support) varies with the applications running on the device. The bandwidth requirements usually depend on the type of video service. While a realtime video phone application requires sustained high bandwidth for the session, downloadable video can use any amount of available bandwidth to pre-fetch the video sequences for playback later. Irrespective of the nature of the video service, the playback of the video consumes significant power; the power consumption is determined by the coding algorithms and encoding parameters used. Resource consumption can also be managed by using object-based content representation together with adaptive playback [2, 3]. As the complexity of the coding algorithms increases, typically, resources consumed in playing back the video also increases. At any given bitrate, a video bitstream can be encoded with varying degrees of complexity representing varying quality and resource consumption. Video servers should code content in a way that suits the receiving device and also satisfies user needs. A server has to understand the receiving devices' capabilities and the user needs before the content delivery begins. This implies some form of capability exchange before a session begins. Describing decoding capabilities is difficult and has to be specified in more detail than just indicating the coding algorithm used. Furthermore these decoding capabilities vary and a receiver has to reevaluate its capability before each session or even during an ongoing video session. In this paper, we analyze the complexity and complexity description of the H.264 video and develop metrics that can be used to communicate bitstream complexity as well as receiver capabilities. We show through experimentation that the metrics are reliable

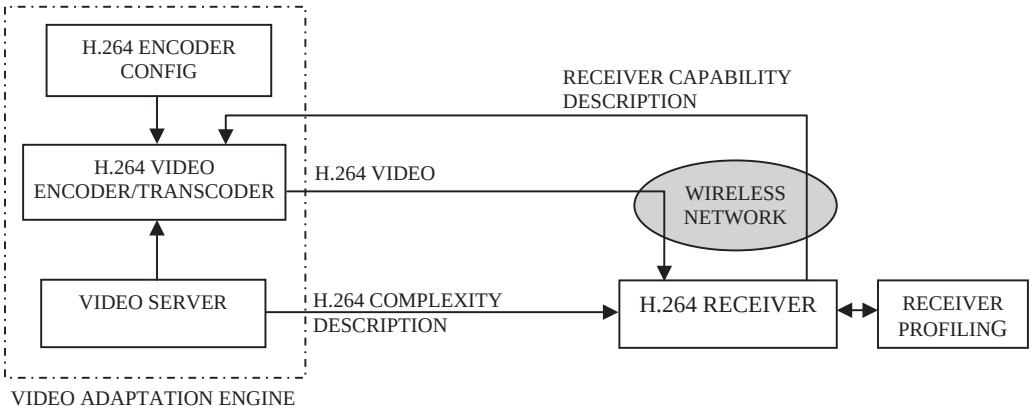

FIGURE 1. Complexity estimation to enable adaptive H.264 encoding.

and represent the actual complexity of the coded video. We focus primarily on the decoding complexity studying the complexity added by each of the encoding tools used. Based on such knowledge, a server can select a set of encoding tools that meet the decoder capabilities while maximizing quality. An application scenario involving resource adaptive encoding is shown in Figure 1. During session setup, a receiver is provided with the complexity description of an encoding of the requested content. If the receiver does not have enough resources to process the content with the given complexity, the receiver can request the sender for a bitstream with acceptable complexity by communicating its capability description. The exchange of receiver capability description and the complexity estimates of the encoded bitstream facilitate adaptive encoding and transcoding that meet the receiver capabilities. The receiver capabilities should be described in terms meaningful to the sender. For example, a receiver capability description that indicates a 200 MHz XScale process is of little use to the sender. The sender has to know the types of bitstreams the receiver can handle with the currently available resources. Bitstream complexity can be used for this purpose; i.e. a receiver describes its capabilities in terms of the bitstreams it can process with the currently available resources.

The rest of the paper is organized as follows. An overview of the H.264 video coding standard is provided in Section 2. Section 3 presents bitstream complexity description at various levels of abstraction. Results and discussion are presented in Section 4. Section 5 discusses the future work and conclusions are presented in Section 6.

2. OVERVIEW OF THE H.264 VIDEO CODING ALGORITHM

The H.264 video coding standard has been developed recently through the joint work of the International telecommunications union's video coding experts group and the International organization for standardization's motion pictures experts group (MPEG) [4, 5, 6]. The standard is also referred to as MPEG-4 Part 10 or MPEG-4 Advanced Video Coding (MPEG-4 AVC). To avoid any confusion, we use MPEG-4 video to refer to part 2 of the MPEG-4 standard and use H.264 to refer to the H.264/MPEG-4 AVC video coding standard. The H.264 video coding standard is flexible and offers a number of new tools to support a range of applications with very low as well as very high bitrate requirements. Compared to MPEG-2 video, the H.264 video format gives perceptually equivalent video at 1/3 to 1/2 of the MPEG-2 bitrates. The bitrate gains are not a result of any single feature but a combination of a number of encoding tools.

The dramatic bandwidth savings will encourage users, such as TV broadcasters, to start adopting the technology as they can now use the bandwidth savings to provide new channels or new data and interactive services. With the coding gains of H.264, full length high definition television (HDTV) resolution movies can now be stored on DVDs. Furthermore, the fact that the same video coding format can be used for broadcast TV as well as Internet and mobile video streaming will speed up the adoption of H.264 video.

The H.264 video uses the same hybrid coding approach used in the other MPEG video standards: motion compensated transform coding. This hybrid coding approach uses transform coding to exploit spatial redundancies, motion compensation to exploit temporal redundancies and variable length coding (VLC) to exploit statistical redundancies. The motion compensation process in H.264 supports a generalized multi-frame prediction resulting in improved prediction efficiency. Figure 2 shows the block diagram depicting the main components of an H.264 video encoder. The main components as shown in the diagram are: (i) an integer transform (T) with energy compaction properties similar to that of the discrete cosine transform (DCT), (ii) an in-loop de-blocking filter (DF) to reduce block artifacts, and (iii) intra frame prediction (IFP) and (iv) the motion compensation engine (ME and MC). The coder control operation is responsible for functions such as reference frame management, coding mode selection, and managing the encoding parameter set. The H.264 standard introduces several other new coding tools that improve coding efficiency. We discuss some of these tools below. Further information on the H.264 video coding standard can be found in [5].

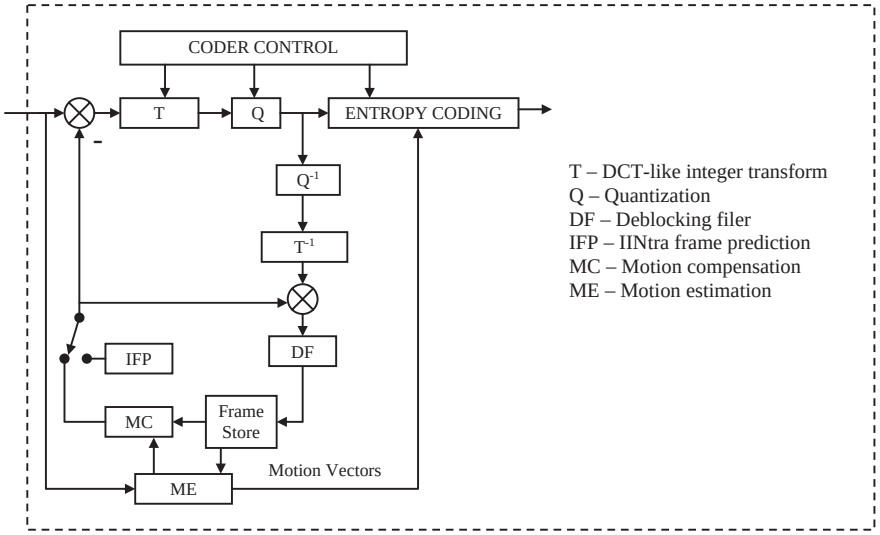

FIGURE 2. Components of an H.264 video encoder.

TABLE 1. Chroma formats and sampling.

Chroma_Format_idc	Chroma Format	SubWidthC	SubHeightC
0	monochrome	–	–
1	4:2:0	2	2
2	4:2:2	2	1
3	4:4:4	1	1

2.1. Picture formats

The H.264 standard supports different color spaces and sampling resolutions. While the first edition of the standard supported only the YCbCr chroma format with 4:2:0 sampling, the fidelity range extensions (FRE) amendment in the 3rd edition of the standard adds 4:2:2 and 4:4:4 sampling as well as new chroma formats including a monochrome format. Higher fidelity samples are supported with up to 12 bits per sample. The sampling for the luma and chroma samples can be selected independently depending on the content characteristics. The chroma format is coded in the bitstream with the syntax element chroma_format_idc. Table 1 shows the chroma formats supported and the sub-sampling of the chroma components compared with the luma component.

H.264 supports both interlaced and progressive pictures. Compared with MPEG-2, the frame/field coding in H.264 is more general. The interlaced and progressive coding mode decisions can be on a frame by frame basis. Macro block (MB) adaptive field-frame coding can be used to code even small regions within a frame as frame or field coded. Typically, field coding is used when there is fast motion and less spatial correlation and frame coding is more suitable when spatial correlation is high and motion activity is lower.

Each picture is partitioned into slices and MBs for coding. An MB is 16×16 block of luma pixels and the associated chroma blocks (e.g. four 8×8 luma blocks and two 8×8 chroma blocks for the 4:2:0 chroma format). A slice contains an integral number of MBs and when adaptive-frame field coding is used, it contains integral number of pairs of MBs. The slice coding type indicates the type of MBs in the slice. An I slice contains only intra (I) MBs, a P slice can contain I and predictive (P) and a B slice can contain I, P, or bi-predictive (B) MBs. H.264 also supports switching I (SI) and switching P (SP) slices intended for use in streaming applications that need dynamic bitstream switching. The SP slices can have I and P MBs and the SI slices have only SI MBs. The coded picture type indicates the type of slices that can be present in the picture. An I picture contains only I slices, a P picture can include I and P slices and a B picture can include I, P, and B slices.

2.2. Intra-frame prediction

Intra-frame prediction in H.264 allows prediction for intra coded MBs. A MB or sub-MBs are predicted from the previously coded neighboring blocks in the same picture. An MB can have 4×4, 8×8, or 16×16 block prediction modes referred to as Intra_4×4, Intra_8×8, and Intra_16×16, respectively. The 4×4 and 8×8 blocks use nine prediction modes and 16×16 blocks have four prediction modes. The chroma blocks are relatively uniform and have four prediction modes similar to the 16×16 blocks. These intra prediction modes emulate directional prediction, and improve the prediction when directional structures are present in the picture. With the intra-frame prediction, the I-pictures can be coded more efficiently compared to MPEG-2, which does not support intra-frame prediction.

The prediction of the pixels in the current block is based on the pixels at the block boundary. For a 4×4 block, the candidate pixels are indicated in Figure 3b. The pixels of a 4×4 block, indicated by lower case letters a–p, are predicted using the weighted average of the candidate pixels at the block boundary. For example, in prediction mode 0, A is the prediction for a, e, i, m, B is the predictor for b, f, j, n, C is the predictor for c, g, k, o and D is the predictor for d, h, l, p.

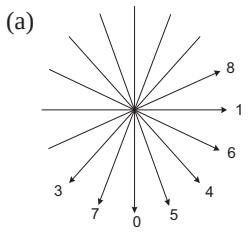

FIGURE 3. (**a**) Directions for prediction modes for 4 × 4 and 8 × 8 blocks. (**b**) Prediction samples for a 4 × 4 block.

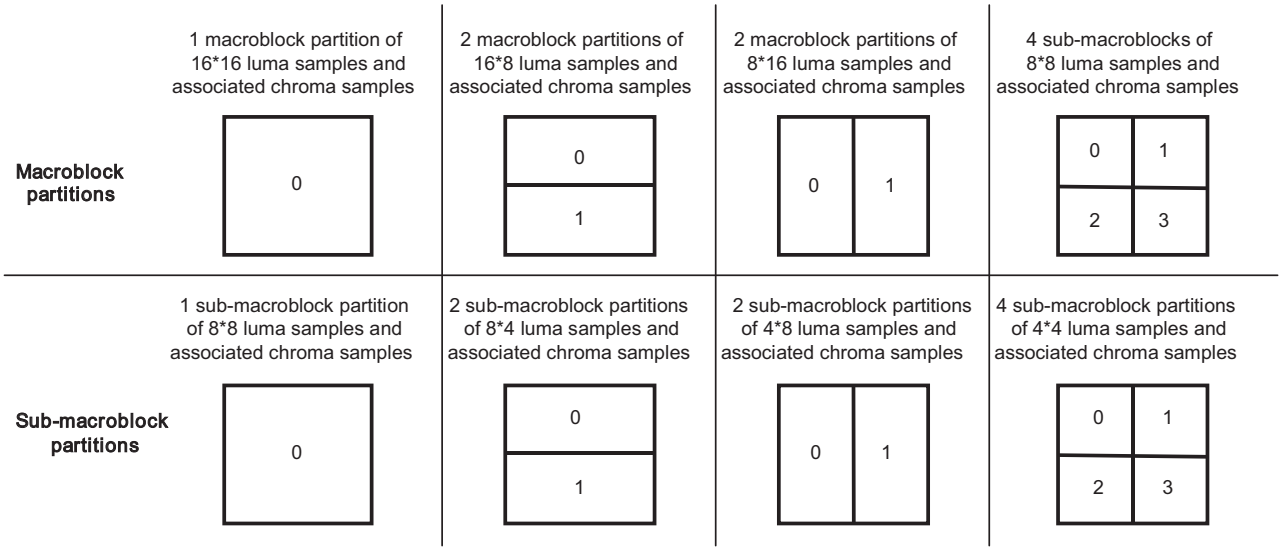

FIGURE 4. Sub-blocks of a macro block in inter prediction.

For prediction mode 5, the predictors are formed using a weighted average of the five candidate pixels A, B, C, J, and M. The directional predictors are supported for 8 × 8 blocks as well. For the 16 × 16 luma blocks and the chroma blocks, four modes are supported. In addition to the modes 0–2 as used for 4 × 4 and 8 × 8 blocks, mode 3, called plane prediction, is used. All the pixels used for intra prediction are the reconstructed pixels prior to applying the deblocking filter.

One drawback of using intra prediction is that intra blocks are now prone to error propagation because of a corrupted prediction. This could occur when error propagates into a P MB because of corruption in the reference frame. The errors from the corrupted P MB will propagate to the intra block when the border pixels from the P MB are used to predict pixels of the intra MB. To avoid this, an encoder can use constrained intra prediction where prediction comes only from intra MB pixels. This constraint makes the prediction inefficient but avoids error propagation.

2.3. Inter-frame prediction

The inter-frame prediction is the traditional motion compensated prediction that is supported in earlier MPEG video coding standards. The H.264 standard extends this in several ways: (i) variable block sizes for motion compensation, (ii) multi-frame references for prediction, (iii) generalized B-frame prediction, (iv) use of B-frames as references, (v) weighted prediction and (vi) fractional pixel accuracy for motion vectors (MVs). These extensions improve the coding performance but also increase the complexity substantially.

The performance of motion compensated prediction typically improves as the predicted block size decreases. With smaller block sizes, the likelihood of finding a closer match increases. However, the disadvantage is that a separate MV has to be transmitted for each of these small blocks. Using smaller blocks gives especially good performance when there is a lot of detail in the coded regions. When the coded regions do not have much variation, a larger block size would suffice. Encoders, typically, evaluate the cost of encoding using different block sizes in the decision making process. A 16 × 16 MB in H.264 can be divided into sub-blocks for motion compensation. Figure 4 shows the MB and sub-macro block partitions. The smallest block that can be used for motion compensation is a 4 × 4 luma block. Motion estimation is typically performed on the luma component and the resulting MVs are scaled down to the chroma resolution. The chroma block is also partitioned based on the partitioning decisions made for the luma component.

The multi-frame prediction allows predictions from multiple reference frames. The number of reference frames that can be used for prediction is up to 16 and the actual number of frames used in a specific sequence is determined by the picture size and the buffer constraints in the profile

and level definitions. To support multi-frame prediction, the encoder and the decoder maintain a list of frames used as reference frames. The P slices use a single list called list 0 and B slices use an additional list called list1. A frame can also be designated as a long-term reference frame by the encoder coding control operations. A reference frame index is used in MB coding to indicate the source of the prediction. The sub-macro blocks of an MB can be predicted from any of the frames in the reference list.

A B slice in H.264 has a generalized definition: a B MB is predicted from any two predictions as opposed to one forward and one backward prediction used in MPEG-2 video coding. Furthermore, a B picture can also be used as a reference picture. Predictions in certain cases can be improved if a weighted prediction is used. This is especially true for scene fade-in, fade-out, and dissolve effects. In weighted prediction, a prediction is multiplied by a weight and an offset added to improve the prediction accuracy. The weighted prediction can be used for both B and P predictions.

The prediction accuracy is also improved by using MVs with fractional pixel accuracy. A quarter-pixel accuracy is supported for luma and a 1/8 pixel vectors are derived for scaled down chroma MVs. The motion estimation and motion compensation complexity increases because of the fractional pixel interpolation. Typically, a motion estimation engine finds a best integer match and then evaluates the half pixels around the integer pixel. The quarter-pixel evaluations are done for the best matched half-pixel vector. The resulting MVs are differentially coded to reduce the amount of bits needed to code the MVs.

2.4. Transform coding

The transform coding is applied to the prediction residual as is done in MPEG-2. The transform used, however, is a 4×4 integer transform instead of the 8×8 DCT traditionally used in many video coding standards. The integer transform is designed such that the transformation involves only additions and shift operations and there is no mismatch between the forward and the inverse transform. This reduces the complexity significantly when compared with the DCT. The smaller size of the transform, compared with the 8×8 DCT in MPEG-2, also reduces the block noise in the decoded video. An 8×8 transform is also added in the 3rd edition of the standard.

The transform coding is applied to the motion compensated residual. A 16×16 MB is divided into 4×4 or 8×8 sub blocks and a 4×4 or 8×8 transform is applied. Irrespective of the block size used for motion compensation, a 4×4 or 8×8 transform is used. For MBs coded in Intra 16×16 mode (intra MB with 16×16 intra prediction), the DC coefficients of the transformed blocks are collected and a hadamard transform is applied to further reduce the correlation. The hadamard transform is also applied to the DC coefficients of chroma blocks. The Intra16×16 prediction is applied when the MB region is relatively uniform. Furthermore, chroma components also have relatively less variations and larger correlation in the DC coefficients of the transformed blocks. Applying the hadamard transform reduces the correlation resulting in lower bitrates. Figure 5 shows the 4×4 transform matrices used in transform coding. The transformed coefficients are scaled using a quantization parameter QP.

$$H_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \quad T_{4\times4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix}$$

FIGURE 5. Transform matrices for 4×4 blocks.

2.5. Deblocking filter

A deblocking filter is applied to the block edges, except picture boundary and block boundaries for which deblocking is turned off. The size of the transform (4×4 or 8×8) determines the block size used for deblocking. Deblocking has the effect of improving the perceptual quality as well as the prediction in inter frames. In each picture, the deblocking is applied to MBs in the raster scan order: left to right and top to bottom. The vertical edges are deblocked first and then the horizontal edges. The deblocking process is applied after all the macroblocks have been reconstructed. This implies that the candidate pixels used in intra prediction are the decoded pixels before the deblocking operation.

2.6. Entropy coding

The entropy coding in H.264 uses universal VLC for all syntax elements except the quantized transform coefficients. The coefficients can be coded using either context adaptive VLC (CAVLC) or context adaptive binary arithmetic coding (CABAC). The CABAC has higher complexity but gives the most coding gains. The universal VLC coding is implemented using exponential Golomb codes. A detailed account of the entropy coding in H.264 can be found in [7].

3. COMPLEXITY ESTIMATION

The complexity as used in this paper refers to the amount of computing resources required to decode a specific piece of content. Content dependencies are inherent in video coding and resources required to encode/decode a video also depend on the content as well as the quality of the video. Complexity description should take these dependencies into consideration and use metrics that are content independent. Furthermore, complexity estimation and description should use metrics that will allow the estimation of the actual computing resources required for specific device architecture. The computing resources required to execute a program can be described in terms of the processor usage, execution time, memory usage, and power consumption. The amount of such resources required varies depending on the underlying hardware architecture. The

main goal of the content complexity description is to allow low-resource terminals such as mobile devices to estimate the amount of resources required and determine whether they can process such content. This allows senders to understand the receiver capabilities and negotiate the delivery of content with acceptable complexity. In this, we focus on the execution time alone.

The profile and level information represent the upper bound on the amount of resources required and do not reflect the actual complexity. The actual complexity is typically much less than the upper bound complexity implied by the particular profile and level used. Furthermore, even when a decoder is designed to process video at a certain profile and level, the current available resources may not be sufficient to process the video. A good estimate of complexity that allows decoders to determine whether they can receive and playback the video with the available resources is necessary. Complexity management is a difficult task and has to be addressed at different levels of abstraction. In our earlier work on MPEG-2 to MPEG-4 transcoding we developed a transcoder with reduced complexity using both the algorithmic optimizations in the transcoding component as well as the optimizations specifically for the target architecture [8]. The processor and system architecture have significant impact on the device capability and resource consumption. The complexity of video coding and the impact on processor architectures is discussed in our earlier work [9]. The complexity analysis of the H.264 video decoding reported by Horowitz *et al*. discusses the implementation complexity of the decoder and does not address the receiver resources and the receiver complexity description [10].

3.1. Describing complexity

Complexity of a compressed video bitstream has to be described in a way that translates it into the resource requirements on a playback device. Video sequences are coded hierarchically and the complexity of a sequence can be described at different levels of the hierarchy. In the following, we describe the hierarchy, complexity estimation at that level of the hierarchy and the usefulness of the metric.

3.1.1. Frames level complexity

A video sequence is coded as a sequence of frames. Individual frames are coded independently (Intra frames) or using a prediction based on the previously coded frames. The temporal resolution of frames is measured using the frame rate which is typically given in frames per second (FPS). The complexity described in FPS indicates the frame rate of a video. For device complexity, it is not possible to indicate the capability meaningfully in terms of FPS. When qualified by the profile and level, for example, a device is capable of processing *N* FPS of video at profile P1 and Level L1, FPS can be used to describe the device capability at a very coarse level. However, the content dependencies are not accounted for and this metric is not an indication of the actual complexity. Resource availability on mobile devices varies dynamically as the other applications, the available battery and the available memory varies. Indicating the complexity with upper bounds is wasteful and will result in devices making incorrect decoding decisions. The FPS metric is also not useful when the sender expects to use adaptive frame rates.

TABLE 2. Execution time in ms for I and P frames in an MPEG-4 decoder.

Bitrate (Kbps)	Time for I frame decoding (ms)	Time for P frame decoding (ms)
384	1060	230
256	980	185
128	900	155

At the same profile and level, and at the same frame rate, a video can have different spatial resolutions and the complexity increases with spatial resolution. The spatial resolution of a video, together with its temporal resolution, gives an improved estimate of the complexity but this is not accurate enough to describe the actual complexity. A device describing its video decoding capability as the ability to decode video of size 352×288 at 15 FPS, is not describing its decoding capabilities properly and devices from two vendors with the same capability description may in fact have completely different actual decoding capability. The primary reason for such inconsistency is that a very high level capability description does not account for the coding modes and content dependencies and require significant context information that describes the device state to be useful.

Another reason FPS is not a meaningful metric by itself is the coding type dependencies. The coding type of a video frame has significant influence on the computing resources required. For example, the I frames of a video sequence typically take more time to decode compared with a P frame and this difference could be substantial on small mobile devices. A video sequence coded at 15 I FPS requires more resources to decode compared with a video sequence with one I and 14 P FPS. Describing the device capability in terms of FPS is thus not very helpful. Table 2 shows the actual decoding times measured for I and P frames in an MPEG-4 decoder running on iPAQ with 233 MHz StrongARM processor.

Given these factors, using frame rate to indicate complexity, without providing substantial contextual information, is not useful.

3.1.2. Slice level complexity

A video frame is typically coded as a set of slices. A slice contains one or more MBs. Depending on the coding algorithm used, the size of a slice and the number of slices per frame can vary. While the slice structure is useful in improving the error resilience of the coded video, it has little or no bearing on the complexity of the video itself.

3.1.3. MB level complexity

Video coding usually processes one block of video at a time and a 16×16 MB is commonly used. An

MB is 16×16 pixels of luminance component and the corresponding chrominance components. The number of chrominance pixels depends on the chroma format used. For example, for 4:2:0 chroma format, an MB of video includes 16×16 luminance component and two 8×8 chrominance components.

Describing the complexity as the MB decoding rate is a better indication of complexity compared with using the frame rate. Since an MB represents a 16×16 block of a frame, decoding an MB reconstructs a 16×16 region of a video frame. The MB decoding rate when qualified with the chroma format used gives a better indication of complexity. This, however, does not address the content dependencies. It is also possible that, because of content dependencies, part of an MB is not coded and hence two coded MBs do not necessarily represent the same amount of compressed bits or the same amount of decoding complexity. The MB rate is given as an upper bound on the decoding rate in the profile and level constraints used in the H.264 standard.

3.1.4. Block level complexity

An MB is usually divided into sub-blocks and the sub-blocks are coded in the final coded representation. The number of sub-blocks depends on the coding standard and the content being compressed. The MPEG-2 and MPEG-4 standards code video use MBs divided into four 8×8 blocks. The H.264 video coding standard allows variable block size for motion compensation but uses 4×4 or 8×8 blocks for the coded video. The coded block rate gives an improved indication of complexity when compared with the MB rate indication.

3.1.5. Coefficients level complexity

A block of size $N \times N$ can at most have $N * N$ coefficients to be coded. In practice, the number of coded coefficients per block is less than $N * N$. Depending on the content dependencies and the quantization applied, the number of coefficients to be coded could be much smaller than $N * N$. For example, an $N \times N$ block of pixels with high spatial redundancy will have very few non-zero coefficients after applying a transform such as DCT. Furthermore, using a large quantization parameter results in fewer non-zero coefficients. Thus the number of non-zero coefficients per block gives the actual number of coefficients that are coded and hence a better estimation of complexity.

Unlike MPEG-4, H.264 supports two types of entropy coding: (i) universal VLC tables and (ii) CABAC. CABAC gives a better compression performance and is more complex. In addition to the number of non-zero coefficients, the entropy coding used should be indicated in the complexity description.

3.1.6. Coding mode

The coding mode has significant influence on the computational complexity. Intra coded blocks do not exploit temporal redundancies and result in larger amounts of coded data. On the other hand, inter coded blocks exploit temporal as well as spatial redundancies and result in smaller amounts of coded data. The inter-coded videos have the complexity

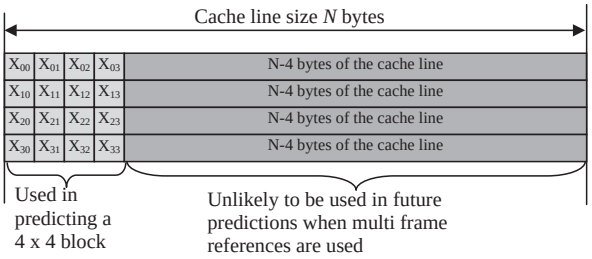

FIGURE 6. Example of cache line reads.

associated with the motion compensation process. The multi-frame motion compensation increases the complexity compared with the motion compensation process in the MPEG-2 video coding.

3.1.7. Memory and motion estimation complexity

The motion estimation process in H.264 supports multiple reference frames. This means two consecutive MBs can use completely different pictures as references. This flexibility in selecting reference pictures results in improved prediction efficiency but increased complexity. The increased complexity is because of increased memory requirements and, more importantly, the likely decrease in cache performance. The cache performance depends on the cache size and the cache model used. Whenever a cache miss occurs, typically one or two cache lines are read into the cache from the main memory. The cache line size is usually 32–128 bytes depending on the architecture and is much larger than the block dimension used in motion compensation. This means that when computing a prediction for a block of data, lot more data than is actually used in computing the prediction is retrieved into the cache. Cache efficiency improves when successive blocks have overlapping predictions that result in a cache hit. Referring to Figure 6, if predicted block Xij is the prediction for block K, the prediction for the subsequent blocks will result in a cache hit if the prediction is from the $4 \times N$ region read into the cache. When multiple reference pictures are used, the additional data retrieved is less likely to be used in computing subsequent predictions. This may even result in cache trashing leading to severe performance casualty. Figure 6 depicts the cache usage for predictions.

When only a single frame is used for prediction, the prediction patterns are regular and can be determined with the knowledge of the maximum search range. This allows cache optimization schemes to be employed. Use of multiple reference frames makes such optimizations difficult and less efficient. Complexity descriptions should take into account the number of reference frames used as well as 'reference frame extent' (for a set of N blocks, how many different reference frames were used). This metric can be used to estimate the cache efficiency.

3.2. Estimating bitstream complexity

The relative complexity of a bitstream can be estimated using the complexity descriptors discussed in Section 3.1. Intra

coded MBs are used for error resilience purposes or when temporal prediction does not give enough bit savings. The 4×4, 8×8 and 16×16 blocks for intra prediction supported by H.264 vary in complexity. A I4×4 block is relatively more complex as it requires more syntactic elements and prediction to be decoded for each of the 16 4×4 blocks. For an I16×16 block, a single prediction mode is used for the entire block. The modes used have an impact on the cache performance. Prediction mode 0 (see Figure 3) is the most cache efficient as all the prediction pixels are likely to be read in a single cache line read. Modes 1 and 2 are the most inefficient as they require 4 (or 16 for a I16×16 MB) cache line reads to read all the neighboring pixels used in the prediction. A higher percentage of I4×4 blocks thus indicates higher complexity.

The complexity of inter-coded MBs is determined by the block size used for motion compensation. An MB coded as 16 4×4 blocks has to decode MVs for the 16 blocks and then form predictions for these blocks. The smaller block sizes are typically used when there is significant detail and variation in the picture. A smaller block size is likely to result in fewer overlapped predictions and may result in cache inefficiencies, but it is also likely to give a better prediction resulting in fewer non-zero coefficients. The number of MVs is also a good indication of the image activity. The length of the MVs typically does not affect the decoding complexity as it represents an offset for the prediction. The actual number of reference frames used may have an impact on systems with low memory and the effects are typically seen in translation lookaside buffer misses and page faults that result in an overall increase in instruction latencies.

The number of non-zero coefficients per MB is also a good estimate of the complexity. Larger number typically means higher complexity. This number is generally high for intra MBs as intra coding does not remove the temporal redundancies. For inter MBs, the prediction performance determines the number of non-zero coefficients. A smaller block size in motion compensation is likely to result in fewer non-zero coefficients but larger number of coded MVs. Fewer non-zero coefficients will also result in reduced complexity in the inverse transform computation because of the operations with zeros. A combination of an average number of non-zero coefficients and coded MVs is a good indication of the complexity of the inter pictures.

4. RESULTS AND DISCUSSION

The experiments were performed using the H.264 reference software version JM8.5. The bitstreams were encoded on a standard and PC and the decoding was done on a Dell AXIM X30 with PocketPC 2003. The JM8.5 decoder was ported to the PocketPC 2003 platform and built using Embedded Visual C++ 4.0. The changes made to port the code as well as to collect the data have no measurable impact on the decoding performance. The decoded YUV output was disabled to eliminate the effects of the time consuming file I/O. The data collected is tabulated in the following tables: the bitstreams used in the experiments are listed in Table 3, Table 4 shows the detailed frame statistics for bitstream no. 1 for the first 5 frames of the bitstream, Table 5 gives the average statistics for bitstreams and Table 6 gives the MB type statistics as a percentage of MBs.

TABLE 3. Bitstreams used in the experiments.

BS no.	Type	Res	No. Ref.	MV range	Frame rate	Bitrate (Kbps)
1	IP	CIF	1	16	15	256
2	IP	CIF	1	16	15	150
3	IP	CIF	1	16	15	100
4	IP	CIF	10	16	15	256
5	IP	CIF	1	32	15	256
6	IP	CIF	1	2	15	256
7	I	CIF	0	0	15	270
8	I	CIF	0	0	15	144

Table 3 lists the bitstreams used and the high level bitstream properties. These high level properties indicate complexity to a certain extent. The bitstream parameters were selected to estimate the effects of MVs, variable block sizes, frame types, and the number of reference frames. Additional work is necessary to study the impact of the VLC, variable block motion estimation, transform size and the use of long term references.

Table 4 shows the details of the data collected for each decoded bitstream. The data includes, for each frame, total coded intra MBs, inter MBs, coded coefficients, coded MVs, and the decoding time in ms. The types of intra MBs, inter MBs and the sub-block types for inter 8×8 MBs is also shown in the table.

The averages for bitstreams are recorded in Table 5. The table also shows an average number of non-zero coefficients per coded MB and an average number of MVs per inter MB. Figure 7 shows the plot of the normalized time complexity (decoding time per frame) together with the normalized values of several high level bitstream metrics. Relative complexity can be estimated by the curves that closely track the time complexity curve. The slope of the curves is sufficient to select a metric that estimates complexity. From Figure 7, the coded MBs per frame tracks complexity closely except for the intra-frame only sequences. There are no skipped MBs in intra frames and the coded MB per frame does not reflect the complexity. The average number of MVs per MB gives an indication of the memory activity and cache usage in the decoder. Based on the decoder cache size, the decoder can estimate the cache efficiency for this specific bitstream. The cache also has an impact on power consumption as a significant amount of power consumption is due to memory transactions [11]. The MV range of 2 was used for encoding the bitstream to demonstrate an extreme case. When such a small MV range is used, the motion compensation process is ineffective and a large number of blocks (68%) are coded as intra blocks resulting in a large number of non-zero coefficients (the 13.6/coded MB is the highest of all the bitstreams tested). The complexity due to motion compensation is small because of a smaller

TABLE 4. Decoded bitstream statistics for BS no. 1.

Fr. no.	Skipped MBs	Intra MBs	I4 × 4	I16 × 16	Inter MBs	P16 × 16	P16 × 8	P8 × 16	P8 × 8	S8 × 8	S44 × 8	S8 × 4	S4 × 4	Coeff.	MVs	Time (ms)
1	0	396	204	192	0	0	0	0	0	0	0	0	0	9828	0	520
2	170	18	9	9	208	59	30	37	82	176	64	72	16	1476	705	464
3	153	47	25	22	196	51	23	23	99	220	84	77	15	2360	745	508
4	124	55	28	27	217	58	37	25	97	207	85	73	23	2396	797	537
5	127	85	40	45	184	48	31	29	76	182	62	53	7	2460	608	537
6	127	62	25	37	207	76	30	41	60	127	68	36	9	1758	589	556

TABLE 5. Averages for bitstreams evaluated.

BS no.	Skipped MBs	Intra MBs	I4 × 4	I16 × 16	Inter MBs	P16 × 16	P16 × 8	P8 × 16	P8 × 8	S8 × 8	S4 × 8	S8 × 4	S4 × 4	Coeff./ MB	MVs/ MB	Time (ms)/ frame
1	119.25	106.63	49.00	57.63	170.13	51.50	26.63	25.63	66.38	151.50	56.75	47.25	10.00	10.95	3.27	532.13
2	150.50	114.00	26.63	87.38	131.50	63.63	20.75	22.25	24.88	67.38	17.50	13.50	1.13	6.67	2.16	500.63
3	171.50	118.38	15.00	103.38	106.13	64.63	14.25	16.13	11.13	30.88	8.00	5.25	0.38	5.08	1.74	467.13
4	121.00	158.88	75.63	83.25	116.13	53.00	20.88	22.75	19.50	48.13	15.63	11.88	2.38	12.47	2.18	516.25
5	122.50	104.38	47.75	56.63	169.13	53.25	25.75	28.50	61.63	140.25	53.50	44.38	8.38	10.39	3.14	529.38
6	124.38	186.88	85.00	101.88	84.75	48.50	10.13	10.25	15.88	36.13	13.50	11.00	2.88	13.16	2.19	502.75
7	0.00	396.00	130.75	265.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	11.80	0.00	515.38
8	0.00	396.00	28.38	367.63	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	5.42	0.00	476.63

TABLE 6. Percentage of MBs of each type in bitstreams.

BS no.	Intra MB	I4 × 4	I16 × 16	Inter MB	P16 × 16	P16 × 8	P8 × 16	P8 × 8
1	38.53	45.96	54.04	61.47	30.27	15.65	15.06	39.02
2	46.44	23.36	76.64	53.56	48.38	15.78	16.92	18.92
3	52.73	12.67	87.33	47.27	60.9	13.43	15.19	10.48
4	57.77	47.6	52.4	42.23	45.64	17.98	19.59	16.79
5	38.16	45.75	54.25	61.84	31.49	15.23	16.85	36.44
6	68.8	45.48	54.52	31.2	57.23	11.95	12.09	18.73
7	100	33.02	66.98	0	0	0	0	0
8	100	7.17	92.83	0	0	0	0	0

percentage of inter blocks. Describing the complexity, using the average statistics as shown in Table 5, gives a reasonable estimate of the complexity for the decoders to make informed decisions on playing the video.

The MPEG-4 video coding standard specifies a complexity estimation tool to describe bitstream statistics [12]. The MPEG-4 video standard allows for optionally including the statistics of the encoding algorithm, coding modes, and parameters. In practice, complexity estimation is usually disabled as it is not well understood how this description relates to the receiver capabilities and the standard does not specify the statistics most relevant for receiver resource consumption. The H.264 video differs substantially from MPEG-4 and the statistics used in the MPEG-4 video complexity estimation are not relevant and do not cover all aspects of H.264 coding. Using the MPEG-4-like approach and describing all possible statistics in H.264 video coding will result in a very large set of statistics that is not really useful. In this work, we examine the bitstream characteristics that are most relevant to complexity estimation and their reliability in estimating the complexity. Keeping the complexity metrics to a minimum will result in a smaller burden on the encoder in collecting the metrics as well as encoding bitstreams conforming to the requested complexity.

5. FUTURE WORK

The complexity description using the averages of coding modes as shown in Table 5 gives a reasonable estimate of complexity. This paper primarily considers decoding time complexity. Two other receiver resources that are significantly affected by the bitstream complexity are memory and power consumption. As systems become more and more memory bound (The CPU is much faster than the memory), the memory latencies will become more significant. This impact could be substantial on mobile devices with small processor cache. Also, a better estimate of memory complexity is necessary. We will explore the idea of estimating the cache efficiency by describing the overlapped regions. We will also examine the effects of multiple reference frames and variable block size on

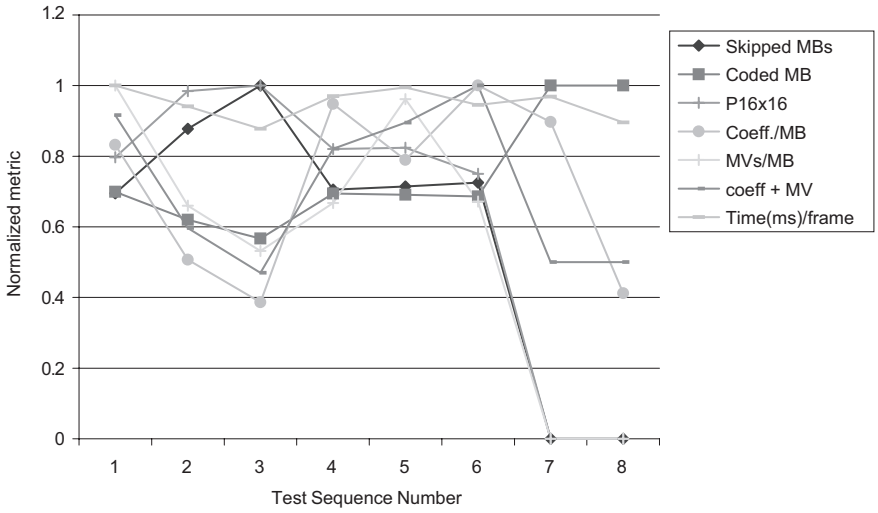

FIGURE 7. Normalized time complexity and bitstream metrics.

low memory mobile devices. Our ongoing work relates to estimating the impact of bitstream decoding on power consumption and cache usage. With increasing use of mobile devices for multimedia applications, power optimized encoding is gaining greater importance.

6. CONCLUSIONS

The H.264 coding standard is a flexible standard with a potential to serve the needs of a range of applications from mobile video services to HDTV broadcasts. The H.264 coding standard performs significantly better than the MPEG-2 standard across the range of bitrates. This improved performance comes at a cost of increased encoding and decoding complexity. This paper presented a general overview of H.264 emphasizing the features that affect the complexity of the decoders. The complexity was analyzed to determine the appropriate level of complexity description for decoders to estimate the amount of resources needed to process the coded video. An H.264 video bitstream complexity can be reasonable estimated using metrics such as average number of coded MBs, coding modes, coded coefficients and coded MVs. The impact of video decoding on system resources such as memory, cache and power consumption has to be studied to get an accurate estimate of complexity and resource usage.

REFERENCES

[1] Kalva, H. (2004) Issues in H.264/MPEG-2 video transcoding. In *Proc. IEEE Consumer Communications and Networking Conf.*, Las Vegas, NV, January 5–8. pp. 657–659. IEEE.

[2] Vetro, A. and Kalva, H. (2003) Technologies and standards for universal multimedia access. In Furht, B. and Marques, O. (eds.), *Handbook of Video Databases*. CRC Press, Boca Raton, FL. ISBN 0-8493-7006-X.

[3] Kalva, H. (2004) Designing object-based audio-visual content representation format for mobile devices. In *Proc. 47th IEEE Int. Midwest Symp. on Circuits and Systems*, Hiroshima, Japan, July 25–28. pp. III-479–III-482.

[4] Wiegand, T., Sullivan, G., Bjontegaard, G. and Luthra, A. (2003) Overview of the H.264/AVC video coding standard. *IEEE Trans. Circ. Syst. Vid.*, **13** (7), 560–576.

[5] ISO/IEC JTC1/SC29/WG11 (2004) *Text of ISO/IEC FDIS 14496–10: Information Technology—Coding of audio-visual objects—Part 10: advanced video coding* (3rd edn). MPEG Document N6540. ISO.

[6] Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T. and Wedi, T. (2004) Video coding with H.264/AVC: tools, performance, and complexity. *IEEE Circuit. Syst.*, **4** (1), 7 – 28.

[7] Marpe, D., Schwarz, H. and Wiegand, T. (2003) Context adaptive binary arithmetic coding in the H.264/AVC video coding standard. *IEEE Trans. Circ. Syst. Vid.*, **13** (7), 620–636.

[8] Kalva, H., Vetro, A. and Sun, H. (2003) Performance optimization of the MPEG-2 to MPEG-4 video transcoder. In *Proc. SPIE*, **5117**, 341–350.

[9] Furht, B. (2000) Processor architectures for multimedia. In Furht, B. (ed.), *Handbook of Multimedia Computing*. CRC Press, Boca Raton, FL, pp. 447–467.

[10] Horowitz, M., Zoch, A. and Kossentini, F. (2003) H.264/AVC Baseline Decoder Complexity Analysis. *IEEE Trans. Circ. Syst. Vid.*, **13**(7), 704–716.

[11] Mizuno, H., Kobayashi, H., Onoye, T. and Shirakawa, I. (2002) Power estimation at architecture level for embedded systems. In *Proc. IEEE Int. Symp. on Circuits and Systems* (ISCAS2002), Arizona, May 26–29. Vol. II, pp. 476–479. IEEE.

[12] ISO/IEC JTC11/SC29/WG11 (2003) *Information technology–Coding of audio-visual objects – Part 2, MPEG-4 video coding* (3rd edn). ISO/IEC 14496–2:2003. ISO.