



166166

1

A Method for Computational Graceful Degradation in an Audiovisual Compression System

This invention is useful in a multimedia encoding and decoding environment where the computational demands for decoding a bitstream is not well defined. It is also useful in cases where channel capacity is limited and some form of quality of service guarantee is required. It is also useful for inter working between two video services of different resolutions.

It is common in the case of software decoding to employ some form of graceful degradation when the system resources is not sufficient to fully decode all of the video bitstream. These degradation ranges from partial decoding of the picture elements to dropping of complete pictures. This is easy to implement in the case of a single video stream.

In the proposed new ISO/IEC SC29/WG11 standard of MPEG-4, it is possible to send multiple Audiovisual, AV, objects. Therefore, the total complexity requirements no longer depend on one single stream but on multiple streams.

In compression systems such as MPEG-1, MPEG-2 and MPEG-4, a high degree of temporal redundancy is removed by employing motion compensation. It is intuitive to see that successive pictures in a video sequence will contain very similar information. Only regions of the picture that are moving will change from picture to picture. Furthermore, these regions



G00002

usually move as a unit with uniform motion. Motion compensation is a technique where the encoder and the decoder keep the reconstructed picture as a reference for the prediction of the current picture being encoded or decoded. The encoder mimics the decoder by implementing a local decoder loop. Thus, keeping the reconstructed picture synchronized between the encoder and decoder.

The encoder performs a search for a block in the reconstructed picture that gives the closest match to the current block that is being encoded. It then computes the prediction difference between the motion compensated block and the current block being encoded. Since the motion compensated block is available in the encoder and the decoder, the encoder only needs to send the location of this block and the prediction difference to the decoder. The location of the block is commonly referred to as the motion vector. The prediction difference is commonly referred to as the motion compensated prediction error. These information requires less bits to send than the current block itself.

In intra-picture coding, spatial redundancy may be removed in a similar way. The transform coefficients of the block can be predicted from the transform prediction of its neighbouring blocks that have already been decoded.

There are two major problems to be solved in this invention. The first is how to indicate the decoding complexity.

requirements of the current AV object. In the case where there are multiple AV objects, the systems decoder must decide how much resource should be given to a particular object and which object should have priority over another. In other words, how to model the complexity requirements of the system. A point to be noted here is that the complexity requirements of the decoder is dependent on the implementation of the decoder. An operation that is complex for one implementation may be simple for another implementation. Therefore, some form of implementation independent complexity measure is required.

The second problem is how to reduce complexity requirements in the decoder. This deals with the method of reducing the complexity requirements of the decoding process while retaining as much of the information as possible. One biggest problem in graceful degradation is the problem of drift caused by errors in the motion compensation. When graceful degradation is employed the reconstructed picture is incomplete or noisy. These errors are propagated from picture to picture resulting in larger and larger errors. This noise propagation is referred to as drift.

According to the invention in a first aspect, there is provided a method for encoding a visual object, comprising:

- encoding a visual object to obtain compressed coded data, wherein the compressed coded data is obtained through at least one of a plurality of different operations which
- 5 can be performed on a block by block basis,
- generating a descriptor capable of including information of a respective parameter related to the number of times that each of said operations was performed in a unit of blocks, said descriptor associated with a complexity of decoding the compressed coded data; and
- 10 multiplexing the descriptor with the compressed coded data.

According to the invention in a second aspect, there is provided, a method for encoding a visual object having a plurality of pictures, comprising:

- encoding a visual object to obtain compressed coded data, wherein the
- 15 compressed coded data is obtained through segmenting each picture in a plurality of blocks and through performing at least one of a plurality of different operations on a block by block basis,
- generating a descriptor related to the number of times that each of said operations was performed during encoding said picture, said descriptor associated with a complexity of
- 20 decoding the compressed coded data; and
- multiplexing the descriptor with the compressed coded data.

Preferably the operations of each aspect include at least one of: a block entropy encoding operation; a block motion compensation operation; a block quantization

- 25 operation; and a block transforming operation.

In order to solve the problems the following steps are taken in the described embodiment of the current invention.

The AV object encoder encodes the AV object in a manner that would allow different amounts of graceful degradation to be employed in the AV object decoder. Parameters relating to the computational complexity requirements of the AV objects are

transmitted in the systems encoder. Implementation independent complexity measure is achieved by sending parameters that gives an indication of the operations that are required.

At the systems decoder, estimates of the complexity required are made based on these parameters as well as the implementation methods being employed. The resource scheduler then allocates the appropriate amount of resources to the decoding of the different AV objects. In the AV object decoder, computational graceful degradation is employed when the resources are not sufficient to decode the AV object completely.

Examples of a method for computational graceful degradation in an audiovisual system will now be described with reference to the accompanying drawings, in which:

Figure 1 is an overall block diagram of the current invention;

Figure 2a shows an example of an encoder of the current invention;

Figure 2b shows an example of a decoder of the current invention;

Figure 3 illustrates the embodiment of the sub-region and the motion vector restriction of the current invention;

Figure 4 illustrates the embodiment for the pan-scan vectors and the sub-region dimensions in the current invention;

Figure 5 illustrates the second embodiment for the padding method of the motion compensated prediction at the sub-region boundary; and,

Figure 6 illustrates the block diagram for the Complexity Estimator.

Figure 1 shows an overall system block diagram of the current invention. The encoder module, 110, encodes the video sequence to allow computational graceful degradation techniques. The output of the encoder is a coded representation of the video sequence that is placed in the encoding buffer, 120. At the same time the video sequence and the coded representation are also passed to the complexity parameter encoder, 130 where the parameters associated with the operation that are required for decoding is computed and encoded. These information together with the output of the encoding buffer are passed to the System Encoder and Multiplexer module, 140, where the system multiplexed stream is formed and transmitted through the transmission media, 150. The Demultiplexer and System Decoder module, 160, then receive the system-multiplexed stream, where the bitstream is demultiplexed into its respective elementary streams. The video elementary stream is passed to the Decoding Buffer, 170, and the complexity parameters are passed to the Scheduler and Complexity Estimator module, 180. From the Decoding Buffer the video elementary stream is passed to the Decoder module, 180. The decoder waits for the commands coming from the Scheduler module before decoding. The Complexity Estimator gives the amount of decoder computational graceful

degradation that is to be employed. Computational graceful degradation is achieved in the decoder by decoding only a sub-region of the complete picture that is deemed to contain the more important information. The encoder will have to prevent the encoder and decoder from drifting apart under these conditions. After decoding, the decoder module also feedback information to the Scheduler and Complexity Estimator so that the information may be used to estimate the complexity of the next picture.

The following is the embodiment of the various modules illustrated in the above invention, 100.

Figure 2a is a block diagram of the encoder in the current embodiment. The input picture is segmented into blocks for processing. Temporal redundancy is removed from the picture by subtracting the motion compensated picture of the previous picture from the current picture. The prediction difference is then transformed into the DCT domain in the DCT module, 111. The resulting DCT coefficients are then quantized in the Quantization module, 112. The quantized coefficients are then entropy coded in the Variable Length Coding (VLC) module, 113, to form the compressed output bitstream. The encoder also has a local decoder loop comprising of an Inverse Quantization module, 114, an Inverse DCT module, 115, a Frame Store, 116, and a Motion Compensation module, 117. The local decoder loops mimics the decoder operations by inverse quantizing the coefficients and transforming it back into the spatial domain

in the Inverse Quantization and Inverse DCT modules respectively. The output is then added to the output of the Motion Compensated module to form the reconstructed picture. This picture is stored in the Frame Store for motion compensation of the next picture. In this invention the encoder modules of Motion Estimation, 118, and Motion Compensation, 117, are changed so that computational graceful degradation may be performed in conjunction with the motion compensation without causing drift.

Figure 3 illustrates the current invention, which segments the picture into two parts. The first part is a sub-region, 220, that must be decoded in the decoder regardless of whether computational graceful degradation is employed or not. The second part is the region outside of the sub-region, 210, which may be discarded by the decoder when computational graceful degradation is employed. Figure 3 also show two blocks that are used for motion compensation. When motion compensation is performed on a block that resides in the sub-region, 250, then the motion compensated prediction block must also come from within the sub-region of the reference picture. In other words the motion vector pointing out of the region, 260, is not allowed. This is referred to restricted motion vector. On the other hand, when a block resides outside the sub-region, 230, the motion compensated prediction block can come from anywhere in the reference picture. This is the same as where there is no sub-region.

Figure 4 shows the embodiment of how to indicate the sub-region within each picture. In order to specify the rectangular sub-region for each picture the following parameters must be specified for each picture and be encoded in the picture header of the compress bitstream. In Figure 4, the picture, 310, and the sub-region, 320, is illustrated. The horizontal offset of the left edge of sub-region from the left edge of the picture, 330, and the vertical offset of the top edge of the sub-region from the top edge of the picture, 340, are shown. These two parameters, referred to as the pan scan vectors, are used to indicate the location of the sub-region. The width, 350 and the height, 360, of the sub-region are the second set of parameters that are required to specify the dimensions of the sub-region.

In a second embodiment of this invention, the motion vector for a block in the sub-region need not be restricted. It is allowed to point out of the sub-region of the reference picture. However padding is required. This is illustrated in Figure 5. Again 410 and 420 indicates the picture and the sub-region, respectively. The motion compensated prediction, 430, is shown straddling the boundary of the sub-region. The portion of the block residing outside of the sub-region, 431, is not used for prediction and is padded by repeating the value of the pixel found at the edge of the sub-region. The portion of the block residing in the sub-region is used without any padding. The similar padding method is used for the rows and columns for blocks located at the vertical edge and horizontal

edge, respectively. Like the first embodiment, this would also enable computational graceful degradation methods to discard the portion of the picture outside the sub-region without causing the encoder and decoder to drift apart.

Apart from motion compensation that may cause drift in inter blocks, intra blocks at the top and left boundary of the sub-region are also restricted from using any blocks outside of the sub-region for prediction. This is because in the computational graceful degraded decoder, these blocks would not be decoded and thus the prediction cannot be duplicated. This precludes the commonly used DC and AC coefficient prediction from being employed in the encoder.

Figure 2b illustrates the Decoder block diagram. The embodiment of the decoder employing computational graceful degradation is described here. The compressed bitstream is received from the transmission and is passed to the Variable Length Decoder module, 191, where the bitstream is decoded according to the syntax and entropy method used. The decoded information is then passed to the Computational Graceful Degradation Selector, 192, where the decoded information belonging to the sub-region is retained and the decoded information outside of the sub-region is discarded. The retained information is then passed to the Inverse Quantization module, 193, where the DCT coefficients are recovered. The recovered coefficients are then passed to the Inverse DCT module, 194, where the coefficients are transformed back to the

spatial domain. The motion compensated prediction is then added to form the reconstructed picture. The reconstructed picture is stored in the Frame Store, 195, where it is used for the prediction of the next picture. The Motion compensation module, 196, performs the motion compensation according to the same method employed in the encoder.

In the first embodiment of the encoder where the motion vector is restricted, no additional modification is required in the decoder. In the second embodiment of the encoder where the motion vector is not restricted, the motion compensation method with padding, 400, as described in the previous section is used in the decoder. Finally, intra blocks at the top and left boundary of the sub-region are also restricted from using any blocks outside of the sub-region for prediction. This precludes the commonly used DC and AC coefficient prediction from being employed.

In this embodiment the Complexity Parameter Encoder consist of a counting module that counts the number of block decoding operations that are required. The block decoding operations are not basic arithmetic operations but rather a collection of operations that are performed on a block. A block decoding operation can be a block inverse quantization operation, a block inverse DCT operation, a block memory access or some other collection of operations that perform some decoding task on the block by block basis. The Complexity Parameter Encoder counts the number of blocks that require each set of operations

and indicate these in the parameters. The reason block decoding operations are used instead of simple arithmetic operations is because different implementations may implement different operations more efficiently than others. There is also a difference in decoder architecture and different amounts of hardware and software solutions that makes the use of raw processing power and memory access measures unreliable to indicate the complexity requirements. However, if the operations required are indicated by parameters that counts the number of block decoding operations necessary, the decoder can estimate the complexity. This is because the decoder knows the amount of operations required for each of the block decoding operations in its own implementation.

In the embodiment of the System Encoder and Multiplexer, the elementary bitstream are packetized and multiplexed for transmission. The information associated with the complexity parameters is also multiplexed into the bitstream. This information is inserted into the header of the packets. Decoders that do not require such information may simply skip over this information. Decoders that require such information can decode this information and interpret them to estimate the complexity requirements. In this embodiment the encoder inserts the information in the form of a descriptor in the header of the packet. The descriptor contains an ID to indicate the type of descriptor it is followed by the total number of bytes contained in the descriptor. The rest of the descriptor contains the parameter for each of the block

decoding operations. Optionally the descriptor may also carry some user define parameters that are not defined earlier.

In the Scheduler and Complexity Estimator, 180, in Figure 1, the time it takes for decoding all the audiovisual objects is computed based on the parameters found in the descriptor as well as the feedback information from the decoder. An embodiment of the Complexity Estimator is shown in Figure 6. The block decoding operation parameters, 181a, 181b and 181c, are passed into the complexity estimator, 183, after being pre-multiplied with the weightings, 182a, 182b and 182c, respectively. The complexity estimator then estimates the complexity of the picture to be decoder and passes the estimated complexity, 184 to the decoder. After decoding the picture the decoder returns the actual complexity, 185, of the picture. The error in the complexity estimation, 186, is obtained by taking the difference between the estimated and the actual complexity of the picture. The error is then passed into the feedback gain module, 187, where the corrections, 188a, 188b and 188c, to the weightings are found. The weights are then modified by these corrections and the process of estimating the complexity of the next picture continues.

The effect of this invention is that the need for implementations that can handle the worst case is no longer necessary. Using the indications of computational complexities and the computational graceful degradation methods simpler decoders can be implemented. The decoder would have the

capabilities to decode most of the sequences but if it encounters some more demanding sequences, it can degrade the quality and resolution of the decoder output in order to decode the bitstream. This invention is also useful for inter working of services that have different resolutions and/or different formats. The sub-region can be decoder by the decoder of lower resolutions where as the decoder of higher resolutions can decode the full picture. One example is the inter working between 16:9 and 4:3 aspect ratio decoders.

CLAIMS

1. A method for encoding a visual object, comprising:
encoding a visual object to obtain compressed coded data, wherein the
5 compressed coded data is obtained through at least one of a plurality of different
operations which can be performed on a block by block basis,
generating a descriptor capable of including information of a respective
parameter related to the number of times that each of said operations was performed in a
unit of blocks, said descriptor associated with a complexity of decoding the compressed
10 coded data; and
multiplexing the descriptor with the compressed coded data.
2. The method of claim 1, wherein the operations include at least one of:
a block entropy encoding operation;
15 a block motion compensation operation;
a block quantization operation; and
a block transforming operation.
3. A method for encoding a visual object having a plurality of pictures, comprising:
20 encoding a visual object to obtain compressed coded data, wherein the
compressed coded data is obtained through segmenting each picture in a plurality of
blocks and through performing at least one of a plurality of different operations on a
block by block basis,

generating a descriptor related to the number of times that each of said operations was performed during encoding said picture, said descriptor associated with a complexity of decoding the compressed coded data; and
multiplexing the descriptor with the compressed coded data.

5

4. The method of claim 3, wherein the operations include at least one of:
- a block entropy encoding operation;
 - a block motion compensation operation;
 - a block quantization operation; and
 - 10 a block transforming operation.