Modeling Subbands of a Wavelet Based Scalable Video Codec

HENDRIK EECKHAUT DIRK STROOBANDT HARALD DEVOS
MARK CHRISTIAENS
Parallel Information Systems (PARIS)
Electronics and Information Systems (ELIS)
Ghent University (UGent)
St.-Pietersnieuwstraat 41, 9000 Gent, Belgium

Hendrik.Eeckhaut@elis.UGent.be http://www.elis.ugent.be/resume

Abstract: In the RESUME project we explore the use of reconfigurable hardware for the design of portable multimedia systems by developing a hardware-friendly scalable wavelet-based video codec. Our scalable video codec provides the ability to decoded the video stream with reduced frame rate, resolution or image quality directly from the original encoded video stream. This is important for portable devices that have different Quality of Service (QoS) requirements and power restrictions.
Our video codec makes use of arithmetic coding to achieve compression by exploiting statistical redundancy in video streams. This technique employs modeling for building its statistical information. In this paper we explore the use of wavelet subband models. We demonstrate that even if the available memory is extremely limited we come very close to the optimal compression.

Key-Words: scalable video, wavelet, arithmetic coding, context modeling

1 Introduction

"Scalable video" is encoded in such a way that it allows to easily change the Quality of Service (QoS) i.e. the frame rate, resolution, color depth and image quality of the decoded video, without having to change the video stream used by the decoder (except for skipping unnecessary blocks of data without decoding) or without having to decode the whole video stream if only a part of it is required.

Such a scalable video codec has advantages for both the server (the provider of the content) and the clients (the consumers). On the one hand the server scales well since it has to produce only one video stream that can be broadcast to all clients, irrespective of their QoS requirements. On the other hand the client can easily adapt the decoding parameters to its needs. A home cinema system can decode the stream at full quality, while a small portable client can decode the stream at low resolution and frame rate without needing the processing power of the larger clients. This way the decoder can optimize the use of the display, the required processing power, the required memory, ...

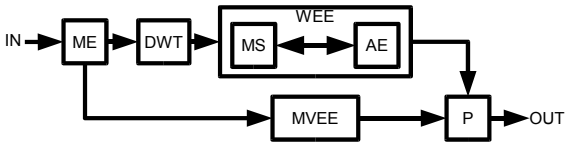

Figure 1: *High-level overview of the video encoder*

The internal structure of one implementation of a scalable encoder is shown in Figure 1 and was described in [1, 2, 3]. It consists of the following parts:

ME: "Motion Estimation" exploits the temporal redundancy in the video stream by looking for

similarities between adjacent frames. To obtain *temporal scalability* (i.e. adjustable framerate of the video), motion is estimated in a hierarchical way as illustrated in Figure 2. This dyadic temporal decomposition enables decoding of the video stream at different bitrates. The decoder can choose up to which (temporal) level the stream is decoded. Each extra level doubles the frame rate.

An intermediate frame is predicted from its reference frames by dividing it into macroblocks and comparing each macroblock to macroblocks in the reference frames. The relative positions of the macroblocks in the reference frames with respect to the intermediate frame are stored as motion vectors. The difference between the predicted and the original frame is called an "error frame".

MVEE: "Motion Vector Entropy Encoder" is responsible for entropy encoding the motion vectors.

DWT: The "Discrete Wavelet Transform" takes a reference or error frame and separates the low-pass and high-pass components of the 2D image as illustrated in Figure 3. Each LL-subband is a low resolution version of the original frame. The inverse wavelet transform (IDWT) in the decoder can stop at an arbitrary level, resulting in *resolution scalability.*

WEE: The "Wavelet Entropy Encoder" is responsible for entropy encoding the wavelet transformed frames. The frames are encoded bitplane by bitplane (from most significant to least significant), yielding progressive accuracy (*quality scalability*) of the wavelet coefficients (Figure 4). The WEE itself consists of two main parts: the "Model Selector " (MS) and the "Arithmetic Encoder" (AE). The MS provides the AE with continuous guidance about what type of data is to be encoded by selecting an appropriate model for the symbol (a bit) that has to be encoded next. It exploits the correlation between neighboring coefficients in different contexts. Finally the AE performs the actual compression of the symbol stream.

P: The "Packetizer" packs all encoded parts of the video together in one bit stream representing the compressed video in such a way that the different types of scalability are fully supported.

Scalability in color depth is obtained by encoding luminance and chrominance information in three different channels in the YUV 4:2:0 format. Omitting the chrominance channels yields a grayscale version of the sequence, allocating more bits to these channels increases the color depth.

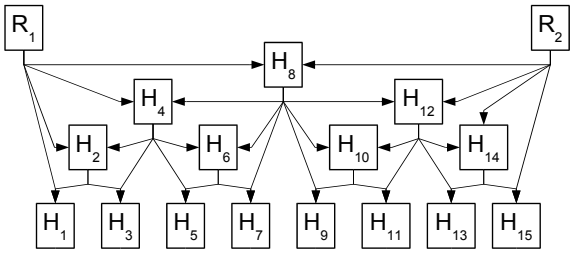

Figure 2: *Framerate scalability. Motion estimation processes one Group of Pictures (GOP) consisting of 16 consecutive frames. The arrows indicate the dependencies between the estimated frames and their references. R_1 is the reference frame of this GOP, R_2 is the reference frame of the next GOP and the H_i are the intermediate frames.*

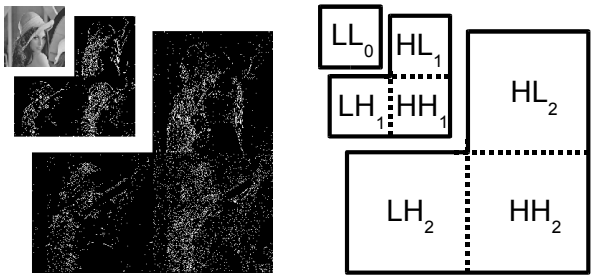

Figure 3: *Resolution scalability. Numbers in subscript reflect the resolution layers.*

The final goal of the RESUME project [4] is to implement a real-time decoder. Therefore we need hard-

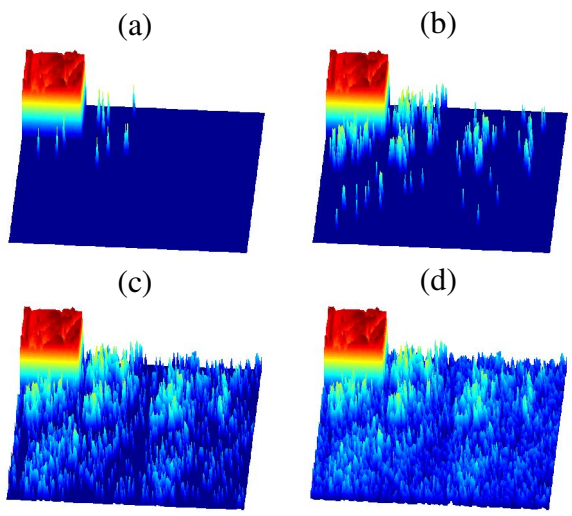

Figure 4: *Quality scalability: decoding more bitplanes of the wavelet transformed image of Figure 3 gives a more accurate wavelet transformed frame.*

```
encode_frame:
  for all subbands:
    load_subbandmodel
    if (LL-subband of reference frame)
      encode (mean of frame, data model)
      subtract mean from resolution level
    encode (number top bitplane, data model)
    for all bitplanes
      encode_bitplane

encode_bitplane:
  for all bits //scanline order
    if coefficient was not significant yet
      if starting bitplane
        encode (bit, toplayer model)
      else
        encode (bit, significance model)
      if bit becomes significant
        encode (sign, sign model)
        update bitmaps
    else //was already significant
      encode (bit, refinement model)
```

Figure 5: *Entropy encoding of one wavelet encoded frame.*

ware acceleration. We target an FPGA implementation to effectively support scalability [5]. Because the WEE gives the most opportunities for improvement we will now focus on this part of the codec.

2 A Hardware-Friendly Wavelet Entropy Decoder

In [3] we proposed a novel approach for coding the entropy of the wavelet transformed frames. In this paper we focus on the remaining research question of how the entropy encoder can be optimally configured.

2.1 The Algorithm

The new algorithm of [3] is shown in Figure 5. All subbands of the wavelet transformed channel are encoded (and decoded) totally independently. This enables the processing of all subbands of the wavelet transformed color channel of the frame in parallel. The subbands are processed bitlayer by bitlayer from top to bottom. The top is the bitplane that contains the most significant bit of the largest absolute value of all coefficients. The bottom is the bitplane containing the least significant bits. The bitlayers are processed in scanline order and all data from one subband is processed sequentially since all bits are now encoded based on information of previously encoded bits.

The compression of the entropy encoder is a result of exploiting statistical dependencies between neighboring pixels. These dependencies are represented by models. A model contains information about the context of the incoming bit and thus about its expected value. Ideally this context would contain all useful information about the neighborhood of the encoded bit. But since this is too expensive, the context is kept small to limit memory accesses.

The Model Selector (MS) (see Figure 6) is the part of the video codec that is responsible for selecting these models. The MS exploits statistical characteristics by encoding bits with a similar distribution using the same arithmetic coder. These characteristics are for example the fact that coefficients become significant (i.e. we encounter its most significant bit) in clusters. For optimal compression, storing all information about previously processed data would be

ideal but since this excludes an efficient hardware implementation only the most relevant information is stored. Our algorithm limits this information to the sign and the significance of each coefficient. This information can easily be organized as two bitmaps with dimensions equal to the subband's. From these bitmaps the number of horizontal, vertical and diagonal significant (or negative) neighbors of the current coefficient are counted to determine the model for the arithmetic coder. In total there are 64 models:

- 1 `data model` that is used to encode data such as the number of the starting bitplane and the mean value of the LL-subband.
- 35 `significance model`s: These models are used to predict the most significant bit of each wavelet coefficient. This group of models is divided in two groups: 8 highest bitplane models and 27 remaining bitplane models. We need these special models for the highest bitplane because there is no information about higher bitplanes when we are encoding this type of bitplane.
- 27 `sign model`s: Depending on the sign of significant neighboring pixels, the sign in the horizontal, vertical and diagonal direction is more likely to be positive or negative.
- 1 `refinement model`, used to encode the refinement bits. These are the remaining bits we come across when processing lower bitplanes than the bitplane where the wavelet coefficient became significant. These bits have the characteristics of noise and are therefore hard to predict.

To determine the models at the borders of the subband, the bitmaps are extended with a symmetric expansion.

2.2 Arithmetic Coder

For the arithmetic coder we opted for a modified version of the CABAC arithmetic entropy encoder used in the AVC codec [6, 7]. This is a low-complexity adaptive, binary arithmetic coder with a probability

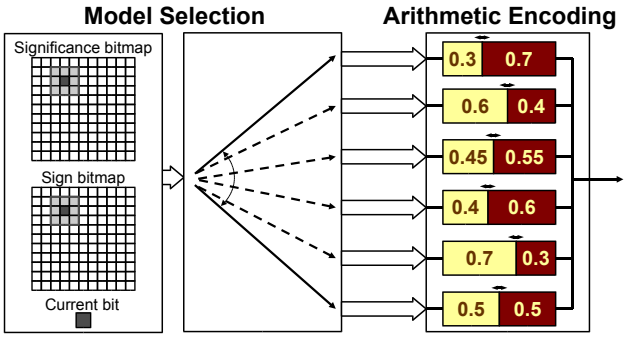

Figure 6: *The model selector selects a model based on information of neighbors, stored in sign- and significance bitmaps. Once the model is determined, the arithmetic encoder encodes the bit with the selected model.*

estimation algorithm that is well suited for an efficient hardware implementation.

We made a few changes to this arithmetic coder to make it a better fit for our wavelet entropy encoder. Since all memories on an FPGA are 9 bit wide, we augmented the 7 bit state per model (i.e. the current estimated probability of the model) to 9 bit. This increased the accuracy for probability estimation and as a consequence the compression performance. We also perfected the transition rule table for updating the probability estimation, but this falls outside the scope of this paper. The fact that only a 9 bit state per model needs to be stored, means that we only require 576 bits for the 64 arithmetic models.

2.3 Initialization of models

Arithmetic coders perform very good if it is possible to accurately estimate the distribution of the incoming bitstream. This is achieved by guiding the arithmetic coder with models, that in the ideal case stand for a certain fixed probability, resulting in near optimal compression. This suggests using lots of models with each their own probability. But if we employ a high number of models, how can the arithmetic coder estimate the probability of the models that are rarely used? We tackled this problem by estimating the probabilities beforehand, by observing the real

probabilities for a set of reference video sequences. By initializing each model with these precalculated values we reach the actual probability much sooner than if we initialized the model conservatively at 0.5.

3 Subband models

There are a lot of different types of subbands which all have distinct statistical properties. First of all there are differences between the LL, HL, LH and HH subbands. In addition, models will be different for subbands of different resolution layers (4 in our case). If we also take the difference between the three color channels and the position in the temporal frame hierarchy into account, we distinguish 480 different types of subband models (for 4 resolution levels).

There are three main approaches in selecting appropriate subband models. One could select the minimal or maximal number of subband models, or could select a well considered subset.

3.1 The maximal case

If each subband type has its own private 64 arithmetic models, there are 30720 different models in total. The cost for using a high number of different subband models is small because we use only 64 arithmetic models at a time. In any case the states of the 64 arithmetic models must be overwritten (reset) every time we start decoding a new subband. So we only have to swap in the appropriate subband model that initializes the 64 models. As a consequence there is no execution cost in using lots of subband models. The only cost is in the storage and fetching of the subband models; the total number of calculations stays the same.

The advantage of using many subband models is that the arithmetic models can be initialized more accurately for each situation since the probability statistics for the different subbands can be predicted separately.

3.2 The minimal case

In the minimal case we use the same set of 64 arithmetic coder models for each subband. This is the optimal case from a storage and caching point of view. But the initialization will of course be much coarser.

Table 1: *Resource requirements of the three cases.*

	Min.	Med.	Max.
temporal classes	1	5	16
color channel	1	2	3
resolution subband	1	10	10
# subband models	1	100	480
# models	64	6400	30720
# bits	576	57600	276480

3.3 The medium case

We can easily reduce the number of subband models compared to the maximal case. Since motion estimation is done in a hierarchical way, all frames of the same (time) level are handled in the same way. A natural choice is to group the sixteen positions in the temporal frame hierarchy per timelevel. This reduces the numbers of subband models by a factor of 5/16. If we also assume that the U and V chrominance channels have very similar statistics, we reduce the number of subband models from 480 to 100. This shrinks the necessary storage memory by a factor 4.8. Other combinations are also possible but are not further explored in this paper. This choice suffices to illustrate the trend.

4 Results

In Figure 7 the average PSNR for decoding 49 frames of the "foreman" sequence (CIF@30Hz) at different bitrates is plotted for the three approaches. The PSNR of each frame is calculated as follows. If $Y_{i,j}$, $U_{i,j}$ and $V_{i,j}$ and $Y'_{i,j}$, $U'_{i,j}$ and $V'_{i,j}$ are resp. the luminance and the two chrominance channels of the original and reconstructed frame of $h \times w$ pixels, then the PSNR is defined as follows:

$$10 \log_{10} \frac{255^2 \frac{3}{2} hw}{\sum (Y - Y')^2 + \sum (U - U')^2 + \sum (V - V')^2} \quad (1)$$

Since the absolute difference in the three cases is nearly invisible in Figure 7, we also plotted the rel-

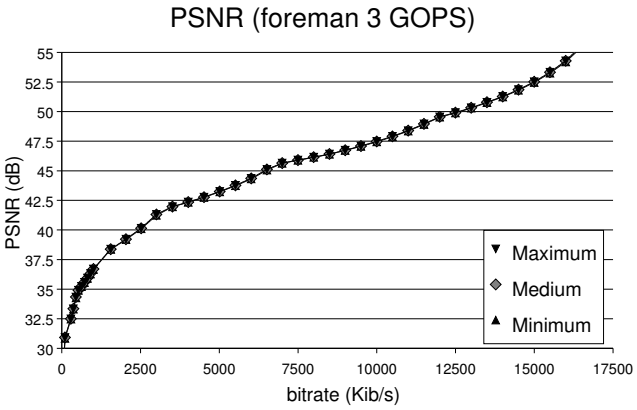

Figure 7: *PSNR vs. bitrate for 49 CIF-frames of the foreman sequence.*

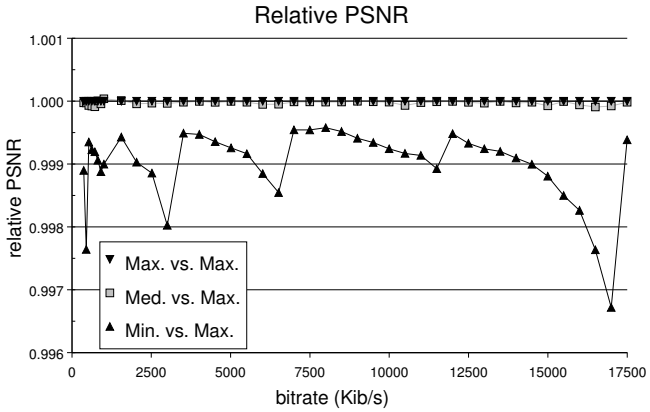

Figure 8: *The relative differences in PSNR of Figure 7.*

ative difference in Figure 8. The maximal case gives the best PSNR but the difference is very small, certainly compared to the medium case. The relative loss in PSNR is on average only about 0.1% when we use only one model.

5 Conclusions

In this paper we investigated the entropy encoding part of our scalable wavelet based video codec and explored the use of wavelet subband models. We can conclude that the deployment of lots of models indeed increases the PSNR but that the absolute gain is very small. Since the overall cost is rather low, the surplus could be justified in most situations. On the other hand, in situations where memory usage is critical, the PSNR penalty for using only one subband model is very limited.

Acknowledgement This research is supported by I.W.T. grant 020174, F.W.O. grant G.0021.03, by GOA project 12.51B.02 of Ghent University and by the Altera University Program. Harald Devos is supported by the fund for scientific research Flanders (F.W.O.).

References

[1] Munteanu A. Wavelet Image Coding and Multiscale Edge Detection - Algorithms and Applications. Ph.D. thesis, Vrije Universiteit Brussel, 2003.

[2] Stroobandt D., Eeckhaut H., Devos H., Christiaens M., Verdicchio F., and Schelkens P. Reconfigurable Hardware for a Scalable Wavelet Video Decoder and Its Performance Requirements. *Computer Systems: Architectures, Modeling, and Simulation*, vol. 3133, July 2004, pp. 203–212.

[3] Eeckhaut H., Devos H., Schrauwen B., Christaens M., and Stroobandt D. A Hardware-Friendly Wavelet Entropy Codec for Scalable Video. In *DATE 2005 Designers' Forum Proceedings*, March 2005.

[4] The RESUME project: Reconfigurable Embedded Systems for Use in Scalable Multimedia Environments. http://www.elis.UGent.be/resume.

[5] DeHon A. The Density Advantage of Configurable Computing. *IEEE Computer*, vol. 33(4), April 2000, pp. 41–49.

[6] Marpe D., Schwarz H., Blättermann G., Heising G., and Wiegand T. Context-Based Adaptive Binary Arithmetic Coding in JVT/H.26L. *Proc. IEEE International Conference on Image Processing (ICIP'02)*, vol. 2, September 2002, pp. 513–516.

[7] Marpe D., Schwarz H., and Wiegand T. Context-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13(7), July 2003, pp. 620–636.