Lefol, D., Bull, D. R., & Canagarajah, C. N. (2006). Performance evaluation of transcoding algorithms for H.264. IEEE Transactions on Consumer Electronics, 52(1), 215 - 222. 10.1109/TCE.2006.1605050

Link to published version (if available):
10.1109/TCE.2006.1605050

Link to publication record in Explore Bristol Research
PDF-document

Performance Evaluation of Transcoding Algorithms for H.264

Damien Lefol, Dave Bull, Nishan Canagarajah

Abstract — *The latest video coding standard H.264 has been recently approved and has already been adopted for numerous applications including HD-DVD and satellite broadcast. To allow interconnectivity between different applications using H.264, transcoding will be a key factor. This paper assesses the performance of existing requantization techniques developed when applied to H.264 together with a new technique. The proposed transcoding algorithm is based on a mixed requantization technique which gives a good compromise between complexity and quality*[1].

Index Terms — Transcoding, Requantization, Bitrate reduction, H.264

I. INTRODUCTION

The development of multimedia systems has had a major influence in the area of image and video coding. This evolution and growth in video application has been concurrent with a massive growth in the communication industry. The wide range of networks available offer different characteristics including range, bitrate and error rate. Distributing new media on these channels, especially video due to its high bandwidth requirements, can be extremely challenging.

Applications using H.264 [1] will range from multimedia content delivery on mobile handsets to High Definition (HD) television broadcasting. To allow such diversity, it will be necessary to have means of adapting the video to the distribution channel. This can be achieved using a transcoder to modify the properties of the encoded bitstream to match as closely as possible the properties of the distribution channel.

Transcoding is required in various applications. For instance, in the case of companies producing video material, it is necessary to change the format depending on the country where the media is broadcast. Consumers also use transcoding in applications such as set top boxes with built in hard drives. Transcoding can also be required if a standard TV is to be used to view HDTV programs. The HD material must first be converted to SD to be displayed.

The objective of bit-rate reduction is to reduce the bitrate while maintaining low complexity and achieving the highest quality possible. Ideally, the reduced rate bitstream should have the quality of a bitstream directly generated at the reduced rate.

Many algorithms have been developed for the requantization of video over the last decade. Some of these have been used successfully in practical applications [2, 3]. It is possible to adapt these algorithms to H.264 but their performances can be variable due to the influence of new features present in H.264.

The most straightforward way to achieve requantization is to decode the video bitstream and re-encode the reconstructed signal at a new rate. Computing new motion vectors from the requantized picture allows a finer approximation for the motion estimation. However this Full Decode and Recode process (FDR) is time consuming and complex. Significant complexity savings can be achieved, while still maintaining acceptable quality, by reusing information contained in the original incoming bitstream [2, 4].

In current video encoders, the predictive encoding scheme is used to reduce the temporal redundancy between consecutive frames. To reconstruct the frame correctly at the decoder, the picture used as reference in the decoder should be exactly the same as that used for prediction at the encoder. Otherwise the mismatched reconstructed picture produces a distortion since the prediction becomes invalid. Since this new distorted reconstructed picture is also used for the future prediction, the distortion error propagates to future frames. Therefore, even a small mismatch can cause significant quality degradation. This error propagation is known as *drift*.

Some requantization algorithms perform bit-rate reduction with no compensation of the errors introduced by requantization [4, 5], whereas others use a closed loop approach to correct those errors [4, 6-8]. The main disadvantage of open-loop algorithms is that they introduce drift in the video sequence. For this reason the two main algorithms used for bit-rate reduction in previous standards (MPEG-2, H.263) were based on a closed-loop algorithm. The first approach, the Cascaded Pixel Domain Transcoder (CPDT) [4], performs the error estimation using the reconstructed picture whereas the second, the Fast Pixel Domain Transcoder (FPDT) [7],uses the residual.

Instead of fully decoding the picture, motion estimation can be done in the transform domain [7]. The requantization error is then computed using only the residual. This FDPT technique is computationally less complex than CPDT as it requires only one frame buffer, one inverse transform and one motion compensation block.

The rest of the paper is organised as follows. Section 2 gives a brief overview of the main requantization algorithms

[1] This work is funded by 3CRL ROAM4G

D. Lefol, D. Bull and N. Canagarajah are with the University of Bristol U.K. (e-mail: Damien.lefol@bristol.ac.uk).

Manuscript received January 6, 2006 0098 3063/06/$20.00 © 2006 IEEE

used with previous standards. The limitations of these algorithms are described and the proposed algorithm is explained in section 3, followed by simulation results in section 4.

II. Requantization Algorithms

A. Cascaded Pixel Domain Transcoder (CPDT)

When adapting the CPDT to H.264 modifications must be made to the original algorithm. First of all the presence of spatial prediction in intra frames is not handled by the original version of CPDT. For this reason the motion compensation (MC) block needs to be modified so that it can work on a MB level instead of a frame level. Another necessary modification is to increase the frame buffer size so that it can handle the greater range of reference frames allowed by H.264. The frame buffer must be able to store at least as many frames as the number of reference frames used in the incoming bitstream. The DCT and inverse DCT blocks have to be replaced by Transform and inverse Transform block. The loop filter block must be added inside both the decoder and encoder part of the cascaded structure. Aside from these changes, the principles of the algorithm remain the same. Figure 1 shows the structure of a CPDT transcoder for H.264.

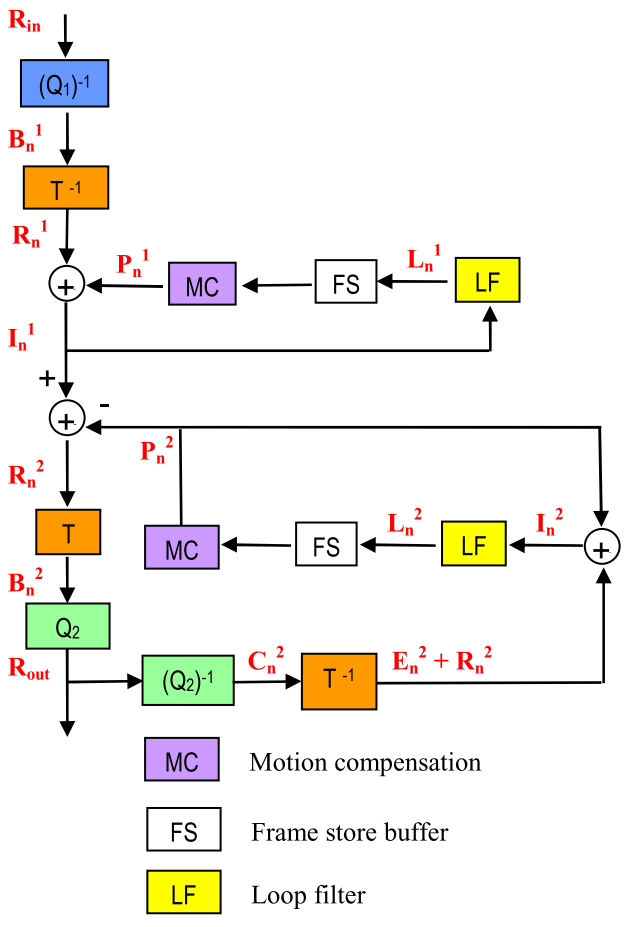


Fig. 1 CPDT for H.264

The relationship between the input and the output of the CPDT transcoder presented in figure 1 can be expressed by equation 1. The output bitstream R_{out} is obtained by requantization of the input bitstream R_{in} to a coarser quantization step Q_2 plus the difference between the values at the output of the motion compensation and loop filter of the decoder and the encoder. This value corresponds to the drift compensation, and is necessary because the picture which will be used for prediction in the final decoder has been modified. Thus, the first part of equation 1 corresponds to an open loop requantization and the second part to the error correction performed by the closed loop system.

$$\begin{aligned} &R_{out} = Q_2\left[T\left[Rq + Drift\right]\right] \\ &with: \quad Drift = MC\left[LF\left[I_{n-1}^1\right]\right] - MC\left[LF\left[I_{n-1}^2\right]\right] \\ &and: \quad Rq = T^{-1}\left[Q_1^{-1}\left[R_{in}\right]\right] \end{aligned} \tag{1}$$

B. Fast Pixel Domain Transcoder (FPDT)

The Fast Pixel Domain Transcoder is one of the most popular requantization techniques used with MPEG-2. The modifications required to adapt the FPDT to H.264 are similar to those done for CPDT. The DCT is replaced by a transform, the motion compensation changed to accept the intra prediction and the frame buffer must be increased to cope with the extra reference frames.

The FDPT technique is computationally less complex than CPDT as it requires only one frame buffer, one inverse transform and one motion compensation block. A fast transcoder algorithm can be derived from the CPDT providing that the clipping functions before the frame memory store are not considered, the motion compensation is a linear operation, motion vectors after transcoding are the same as those before transcoding and no frame-skipping or coding-mode changes are allowed [6, 8].

Using the FPDT with H.264 introduces two new assumptions. The linearity of the loop filter and the introduction of scaling coefficients computed so that $T.T^{-1} = I$.

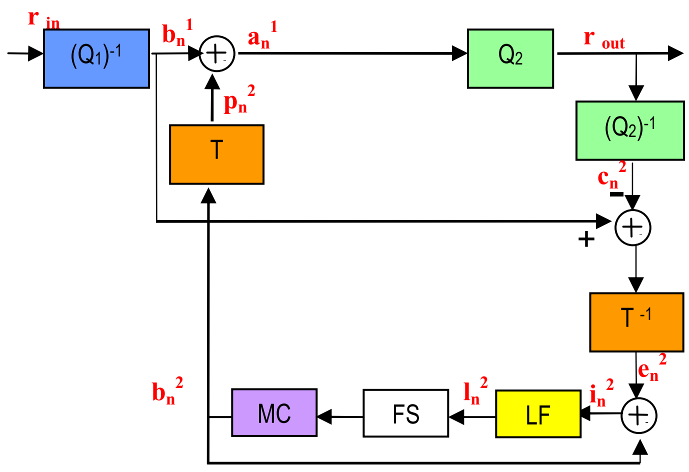


Fig. 2: FPDT for H.264

Equation 2 gives a relationship between the input and the output of the FPDT. It can be seen that the output is linked to the input by the requantization to a coarser quantization step Q_2 and by adding an error correction term which is loop filtered and motion compensated.

$$\begin{aligned} r_{out} &= Q_2\left[Rq + Drift\right] \\ with: \quad Drift &= T\left[MC\left[LF\left[i^2_{n-1}\right]\right]\right] \\ and: \quad Rq &= Q_1^{-1}\left[r_{in}\right] \end{aligned} \qquad (2)$$

If we consider the Motion Compensation (MC), the Loop Filter (LF), the Quantization, inverse Quantization (Q/Q^{-1}) and the Transform, inverse Transform (T/T^{-1}) as linear functions and if moreover we add scaling coefficients so that $T.T^{-1} = I$, equation 1 from the cascaded pixel domain transcoder can be re-written as follows:

$$\begin{aligned} R_{out} &= Q_2\left[Rq + Drift\right] \\ with: \quad Drift &= T\left[MC\left[LF\left[I^1_{n-1} - I^2_{n-1}\right]\right]\right] \\ and: \quad Rq &= Q_1^{-1}\left[R_{in}\right] \end{aligned} \qquad (3)$$

This is the same as the general equation for the fast pixel domain transcoder as we have $R_{in} = r_{in}$ and $I_n^1 - I_n^2 = i_n^2$.

Simulation using FPDT adapted to H.264 shows that it can introduce a severe drift in intra frames. The reason for this drift is that FPDT is based on a mathematical assumption concerning the linearity of functions. Those assumptions have already been proved to be incorrect for MPEG-2 [6, 8] but the drift introduced was negligible. In the case of H.264, the intra prediction process can propagate and accumulate these errors up to 480 times (HD can have 1920 pixels and thus 480 4x4 macroblocks). Moreover, H.264 encoding introduces other sources of errors such as the loop filter and the scaling coefficient used in the transform and quantization [9]. This drift in intra frame can then accumulate in inter frames due to temporal prediction.

The following part discusses the mathematical assumptions made to obtain equation 3. All notation used refers to the one of figure 1 and figure 2.

III. Limitations

A. Non linearity of the motion compensation

To assess the amount of error introduced by the non linearity of motion compensation, we use the following procedure. The first time the CPDT algorithm is activated, the predicted image from the decoder loop of the algorithm is saved along with the one from the encoder loop. The mixed requantization algorithm (MRA) is then activated and the values at the output of the motion compensation are saved. The loop filter is disabled so its non linearity effect is not taken into account. Considering only the second picture of the bitstream (a P frame) we will have access only to one reference frame (the first I frame). As the MRA and CPDT use the same algorithm for intra frames, the values at the input of the motion compensation will be the same except that, for the MRA case, it will be the difference of two frames which is motion compensated instead of two frames ($i_n^2 = I_n^2 - I_n^1$). We can then compare the result of $MC(i_n^2)$ and $MC(I_n^2) - MC(I_n^1)$. If the MC is linear we should have equality. Any difference at the output is due to the non linearity of the motion compensation process. Figure 3 shows the results obtained in the case of a transcoding from QP10 to QP36.

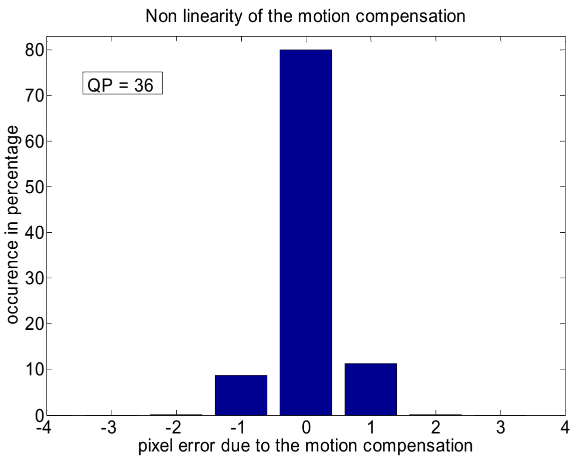


Fig. 3 Occurrence of the error introduced by the motion compensation (QP_2 = 36)

Figure 3 highlights that errors introduced by motion compensation are small, as the maximum error is 2 pixels, and occurs infrequently. An error of one pixel occurs in nearly 20% of the cases, but these errors, even if they tend to accumulate over the GOP are still small and can be neglected. Simulations also show that error occurrence or error values do not change significantly with the quantization parameter. This is logical as errors in motion compensation are introduced by the clipping functions used and thus do not depend on the quantization parameter.

B. Non linearity of the loop filter

To obtain equation 3 from 1 we considered that applying the loop filter to the difference of two pictures was the same as making the difference of two loop filtered pictures. The loop filter is designed to work on the reconstructed values but in the FPDT the values passed to it are residuals. This difference can introduce a severe drift as the behaviour of the filter will be different in the transcoder and in the final decoder. This mismatch is due to the non linearity of the loop filter. There are two main reasons for this non linearity. Firstly, the decision to filter a MB is based on a threshold. It uses values dependent on the macroblock properties, such as the difference between pixels at the border of the macroblock, and the macroblock coding mode decision, such as MB type and motion vectors [10]. The second reason is due to the rounding errors caused by the clipping function used in the loop filter. This clipping was modified in the FPDT algorithm to take into account the fact that residual values are included in a [-128, +128] range whereas reconstructed one are in a [0, 255] range.

To show the non linearity of the loop filter we compare l_n^2 and L_n^1-L_n^2 using a procedure similar to the one previously defined for motion compensation. Before passing in the loop filter we have $i_n^2 = I_n^1 - I_n^2$ without any errors. This means that any error after the loop filter is due to the non linearity or poor adaptation of the loop filter to the filtering of residuals. The results obtained when transcoding a CIF picture from QP 10 to QP 25 and 36 are shown in figure 4 and 5.

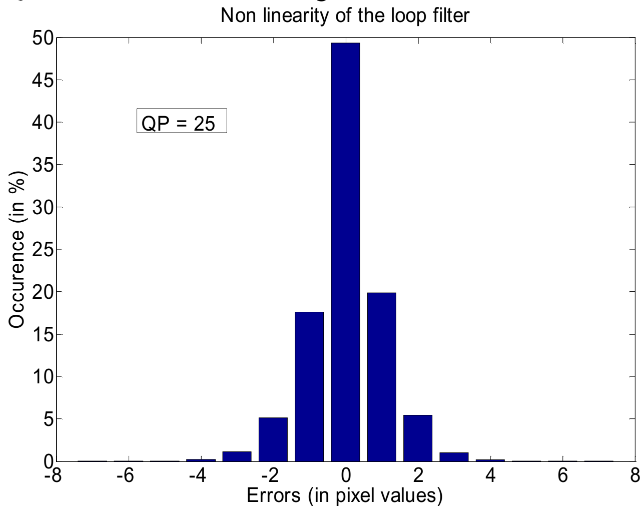


Fig. 4 Repartition of the errors due to the loop filter (QP₂ 25)

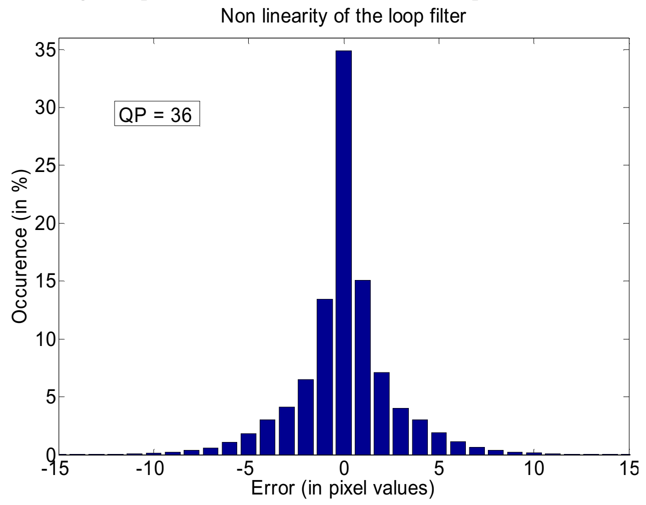


Fig. 5 Repartition of the errors due to the loop filter (QP₂ 36)

It can be noticed that the probability and importance of errors increase with QP. This is logical as the filtering decision and strength is QP dependent. At low QP the loop filter introduces little errors, whereas the amount of error at high QP is large and can be problematic. The larger errors (greater than 2 pixels) are due to the different filtering decision. For instance, the macroblock in the decoder will exceed the threshold and thus be filtered whereas the one in the FPDT will stay below the threshold level and remain unchanged. The small errors are mainly due to the rounding happening in the clipping function.

C. Linearity of the transform process

As the transform is a matrix operation, it is linear and thus it does not introduce errors. Moreover the transform is defined as an integer arithmetic operation so there is no rounding in this process and thus no risk of non linearity. However to obtain the same pixel values after a transform and inverse transform it is necessary to apply a quantization and inverse quantization to the transform coefficients. This is due to the design of the transform and quantization process of H.264 which are not supposed to be performed independently [9, 11]. In FPDT, the transform and quantization need to be performed independently. It is thus necessary to introduce scaling coefficients in the process to take into account the lack of transform or inverse transform. These scaling coefficients have the main inconvenient of being non integer. This means that the transform is no longer an integer arithmetic process and thus the values are prone to rounding errors.

In the transcoder the following scaling coefficients are applied during the quantization and inverse quantization to balance the absence of transform and inverse transform in the fast pixel domain algorithm:

$$r = 0 \Rightarrow scale\_uq = \frac{1}{4} = 0.25$$

$$and \;\; scale\_q = \frac{1}{4} = 0.25$$

$$r = 1 \Rightarrow scale\_uq = \frac{4}{10} = 0.4$$

$$and \;\; scale\_q = \frac{1}{10} = 0.1$$

$$r = 2 \Rightarrow scale\_uq = \frac{1}{2}\sqrt{\frac{2}{5}} = 0.3162$$

$$and \;\; scale\_q = \frac{1}{4}\sqrt{\frac{2}{5}} = 0.1581$$

Where scale_uq represents the scaling coefficient applied when performing an inverse quantization, scale_q the one for quantization and where $r = 0$ for position inside the macroblock $(i, j) \in \{(0,0),(0,2),(2,0),(2,2)\}$ and $r = 1$ for $(i, j) \in \{(1,1),(1,3),(3,1),(3,3)\}$ and $r = 2$ otherwise. Thus for the complete operation the scaling is:

$$r = 0 \Rightarrow scale = scale\_uq \times scale\_q = \frac{1}{16}$$

$$r = 1 \Rightarrow scale = \frac{1}{25}$$

$$r = 2 \Rightarrow scale = \frac{1}{20}$$

A scaling by 64 is also introduced in the process because the inverse quantization includes a multiplication by 2^6 in its coefficients to reduce the rounding errors [12]. To compensate for this in the fast pixel domain transcoder we need to divide by 64 on top of the division by the scale_uq given above to compensate for the lack of inverse transform. It is important to note that the division by 64 is done only in the case of the lack of inverse transform and not for the lack of transform.

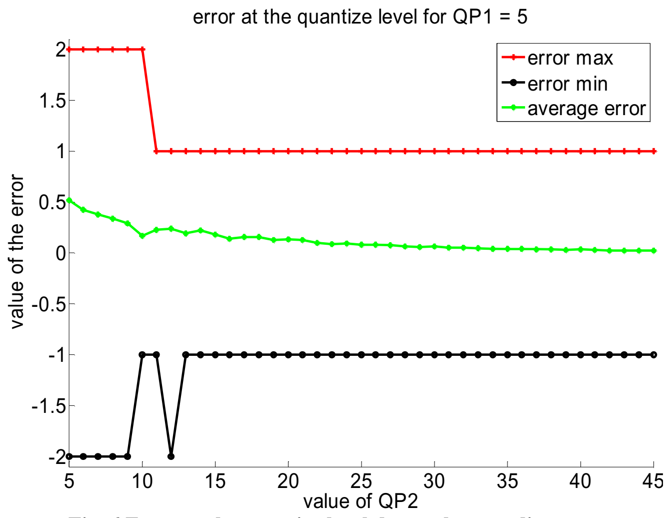


Fig. 6 Error at the quantize level due to the rounding errors

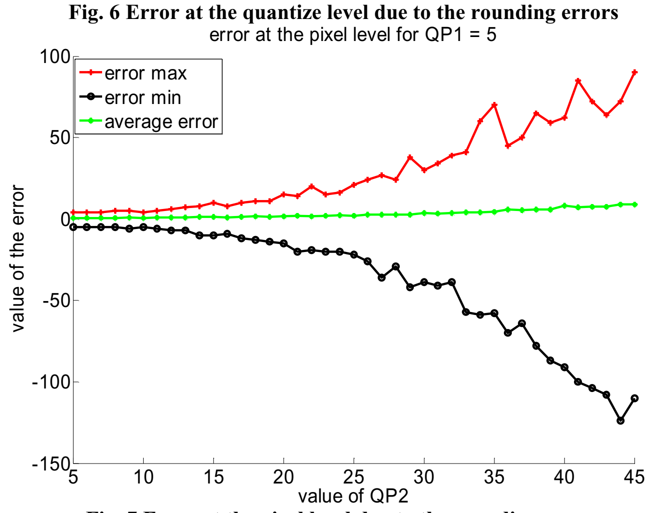


Fig. 7 Error at the pixel level due to the rounding errors

Figures 6 and 7 show the maximum, minimum and average rounding error in the case of a requantization from a QP_1 of 5 to a QP_2 ranging from 5 to 45. It illustrates the influence that rounding errors can have on the final result in the pixel domain. Errors, as they are due to rounding, are very small in the transform domain (maximum of 1 most of the time) but as this error is unquantized, the resulting pixel error can be as high as 100 for large values of QP_2. The average error stays very small but some pixel will have large errors and those errors will propagate and accumulate through drift.

D. Mixed Requantization Algorithm (MRA)

The study of linearity shows we can have large mismatches between the transcoder and the decoder. The simulations performed in section III A, B and C did not take drift into account. When adding the effect of drift, the degradation of quality can be very high. Simulations using FPDT adapted to H.264 show that it can introduce a severe drift, especially in intra frames. Figure 8 shows the difference between an intra frame transcoded using FPDT and the same frame transcoded using CPDT. It highlights the spatial propagation and accumulation of drift inside a frame due to intra prediction. This drift is caused by the non linearity of some functions used in FPDT. In the case of H.264, the intra prediction process can propagate and accumulate these errors up to 480 times.

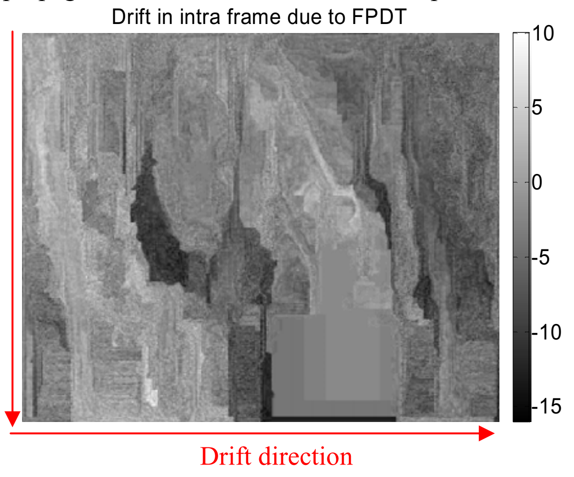


Fig. 8 Drift in intra frames due to FPDT

To avoid this drift, CPDT can be used, although it is computationally more complex. Moreover FPDT can work well in the case of inter frames as temporal prediction introduces less accumulation of errors. Our approach proposed here is a Mixed Requantization Algorithm (MRA) which uses CPDT for the intra frames and FPDT for the inter frames thus combining the advantages of the two different approaches [13]. Moreover using the CPDT for intra frames gives a higher quality for the whole sequence as subsequent inter frames will be predicted from a better quality reference and thus have a higher PSNR.

To make the two algorithms work together it is necessary to change the content of the frame buffer of the pixel domain algorithm before passing it to the transform domain algorithm. This can be done by subtracting the content of the frame buffers one and two of the pixel domain algorithm and put the difference in the frame buffer of the transform domain algorithm. Using parameters described in [14] and [15], our MRA scheme requires 48% less memory than CPDT and 35% fewer computations.

IV. RESULTS

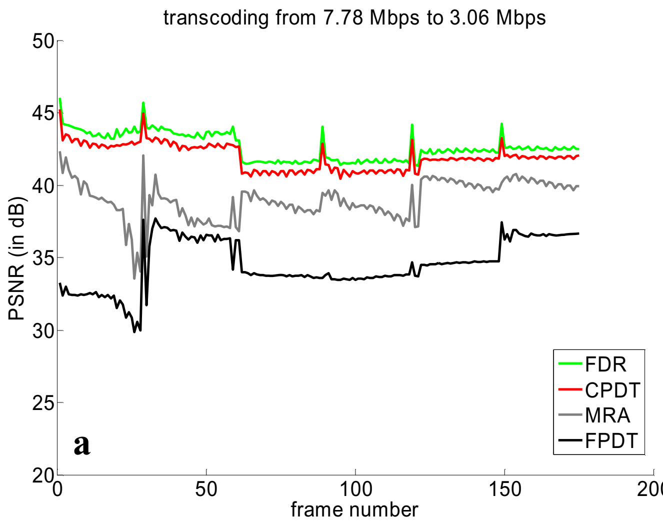
Figure 9 compares the transcoding of a video sequence composed of three concatenated CIF sequences. The first 60 frames are from “Pedestrian”, frames 60 to 120 are from “Tractor” and the last 60 frames are from “Toys”. These sequences were selected to represent a wide range of possible scenarios and concatenated to simulate a normal consumer environment where scene changes will occur regularly. The first sequence contains multiple occlusions and rapid movement, the second a tracking camera and high texture and the third, complex motions and uniform areas. The bitstream has been encoded at 30 frames per second with one intra frame every 30 frames and a group of pictures containing two B frames for every P inter frame.

Four techniques are presented; a full decode and recode (FDR), CPDT, MRA and FPDT. Simulations have been done with an input bitstream encoded with the JM8.5 reference software at a bitrate of 7.78 Mbps and an output bitstream after requantization of 3.06 Mbps.

The plots in figure 9 compare the PSNR values obtained with the four different techniques using two different input bitstreams. The first input bitstream (a) is obtained with an encoder that encodes inter frames using temporal (inter) or spatial (intra) prediction. For the second input bitstream (b) the encoder uses only temporal prediction in inter frames. It shows quite clearly that the CPDT gives far better results than the FPDT and is close to the FDR. Moreover, the FPDT can introduce large changes of quality in the video. These changes are caused by the randomness of the accumulation of the rounding errors. They lead to a flickering video which can be highly uncomfortable for the end-user. The randomness introduced by the rounding errors can create blocking effects (figure 10) in inter frames as two adjacent blocks can be predicted from different reference frames with different rounding errors. This effect cannot be seen in the PSNR values, but it reduces the overall objective quality of the video. The randomness of errors can also be noticed on the rate distortion curves provided in figure 11. The profile of the FPDT curve is very variable and the one of the MRA relatively uneven whereas the one of CPDT and FDR are smooth.

The MRA sequence has a high drift for inter frames in the first sequence of the bitstream containing intra block (frames 1 to 60 of figure 9a). This is due to the video properties. As the first sequence contains occlusions, the encoder uses intra block inside inter frames and thus the accumulation of errors due to the use of FPDT increases. For the rest of the sequence MRA works well.

This type of drift does not exist when the input bitstream of the transcoder is encoded without intra blocks in inter frames (figure 9b). It is however important to note that not using intra blocks in inter frames can reduce the compression efficiency of H.264. Moreover if the transcoder is used in a commercial system it can be uneasy or even impossible to change the encoder settings to avoid intra blocks.



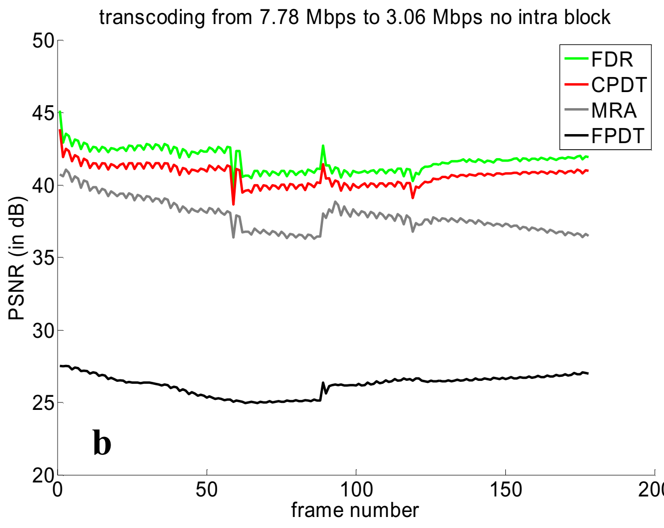


Fig. 9 PSNR comparison for a requantization from 7.78 Mbps to 3.06 Mbps
a. input sequence with intra block in inter frames
b. input sequence without intra block in inter frames

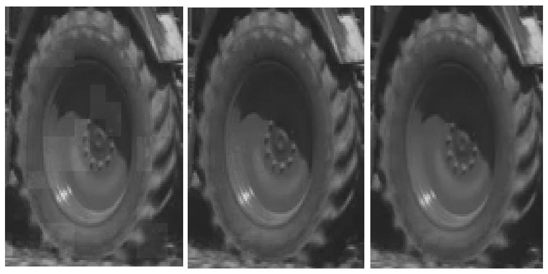


Fig. 10 Blocking effect due to FPDT (left) same frame with MRA (middle) and CPDT (right)

Table 1 shows the average PSNR for the transcoding of the original sequence at different transcoded bitrates. It highlights the fact that as the bitrate decreases, keeping the input encoding decision decreases the efficiency of the compression and thus the quality. This phenomenon is even clearer on the rate distortion curves presented in figure 11. This is due to the large range of compression tools H.264 provides. As the original video has a high bitrate, the encoder uses small macroblock partitions and fine motion vectors. This leads to larger overheads in the bitstream. As the quantization parameter increases, larger macroblock size and coarser motion vectors should be used to take advantage of skip or direct modes which greatly reduce the overheads. However, in the case of a CPDT, where the encoding decisions are kept, this is not possible. Mode refinement can compensate for this, but increases the transcoding time. With FPDT or MRA, mode refinement is not possible as we work on the residual and thus cannot recompute the value of the new predictor if the mode changes.

Bitrate (Mbps)	FDR (in dB)	CPDT (in dB)	MRA (in dB)	FPDT (in dB)
6.39	46.73	46.60	42.28	35.40
4.73	46.61	44.53	41.80	35.93
3.06	42.64	42.06	39.16	34.56
1.32	38.00	36.18	33.61	31.70
0.88	35.82	31.20	28.80	28.18

Table 1 Comparison of the PSNR obtained at different transcoded bitrates

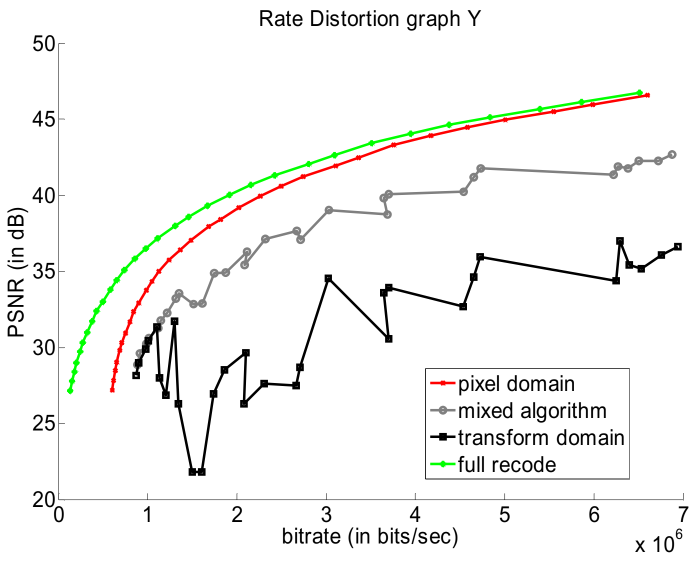


Fig. 11 Rate distortion for the different transcoding strategies

Different mode refinements are possible. It can be motion vector refinement in inter frame [16] or mode changes in intra frame [17]. It is then possible to vary the refinement strategies depending on the computational complexity available to maximize the rate distortion of the transcoded bitstream [18].

V. Conclusion and Future Work

FPDT as developed for previous coding standards cannot be used for H.264 transcoding as it introduces an unacceptable level of drift. The work presented here demonstrates clearly that the drift introduced by this algorithm cannot be ignored with the H.264 standard. A realistic approach for transcoding should be based on CPDT with the possibility of including mode refinement. Only this type of architecture gives a quality suitable for consumer orientated product. In the case of scarce computational power, the MRA is an acceptable alternative even though it can give variable results depending on the video properties and it does not support mode refinement.

The work presented here helps to define the basis for a transcoding platform orientated toward consumer markets. This work could be pursued further by developing suitable refinement strategies to balance complexity and quality.

References

[1]ISO, "ISO/IEC 14496-10 AVC | ITU-T Rec. H.264 International Standard for Advanced Video Coding," 2003.

[2]P. N. Tudor and O. H. Werner, "Real-time transcoding of MPEG-2 video bit streams," presented at Proceedings of the 1997 International Broadcasting Convention, Sep 12-16 1997, Amsterdam, Neth, 1997.

[3]ACTS, "Advanced television at low bit rates and networked transmission over integrated communication systems," *Online: web-site: http://www.bbc.co.uk/atlantic.*, 1996.

[4]A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding architectures and techniques: An overview," *IEEE Signal Processing Magazine*, vol. 20, pp. 18-29, 2003.

[5]H. Sun, W. Kwok, and J. W. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 191-199, 1996.

[6]P. A. A. Assuncao and M. Ghanbari, "Frequency-domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, pp. 953-967, 1998.

[7]P. A. A. Assuncao and M. Ghanbari, "Transcoding of single-layer MPEG video into lower rates," *IEE Proceedings: Vision, Image and Signal Processing*, vol. 144, pp. 377-383, 1997.

[8]J. Youn, M.-T. Sun, and J. Xin, "Video transcoder architectures for bit rate scaling of H.263 bit streams," presented at Proceedings of the 1999 7th International Multimedia Conference - ACM MULTIMEDIA '99, Oct 30-Nov 5 1999, Orlando, FL, USA, 1999.

[9]H. S. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 598-603, 2003.

[10] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 614-619, 2003.

[11] M. Wien, "Variable block-size transforms for H.264/AVC," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 604-613, 2003.

[12] I. Richardson, *H.264 and MPEG-4 video compression: video coding for next generation*: John Wiley and Sons, 2003.

[13] D. Lefol, D. Bull, N. Canagarajah, and F. Rovati, "Performance evaluation of transcoding algorithms for H.264," presented at ICCE 2006 - International Conference on Consumer Electronics, "in press", Las Vegas, USA, 2006.

[14] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 704-716, 2003.

[15] X. Zhou, E. Q. Li, and Y.-K. Chen, "Implementation of H.264 decoder on general-purpose processors with media instructions," presented at Image and Video Communications and Processing 2003, Jan 21-24 2003, Santa Clara, CA, United States, 2003.

[16] J. Youn, M.-T. Sun, and C.-W. Lin, "Motion vector refinement for high-performance transcoding," *Multimedia, IEEE Transactions on*, vol. 1, pp. 30-40, 1999.

[17] D. Lefol, D. Bull, and N. Canagarajah, "Mode Refinement Algorithm for H.264 Intra Frame Requantization," submitted to IEEE International Symposium on Circuits and Systems (ISCAS), Island of Kos, Greece, 2006.

[18] J. Yeh and G. Cheung, "Complexity scalable mode-based H.263 video transcoding," presented at Proceedings: 2003 International Conference on Image Processing, ICIP-2003, Sep 14-17 2003, Barcelona, Spain, 2003.

Damien Lefol received the M.S. degree in electronic from the Institut National Polytechnique de Grenoble, France, and the M.S. (Hons.) in signal processing from the University of Bristol, Bristol, U.K.,

He is currently a Phd student in the Image Communications Group in the Centre for Communications Research at the University of Bristol, Bristol, U.K. His research interests include image and video coding and transcoding.

David Bull is Professor of Signal Processing and Head of the Electrical and Electronic Engineering Department at the University of Bristol. Prior to his current appointment he has been a Systems Engineer at Rolls Royce and subsequently a Lecturer at Cardiff University. He leads the Signal Processing activities within the Centre for Communications Research where he is Deputy Director. He has been a member of the UK Foresight Panel and the Steering Group for the DTI/EPSRC LINK programme in Broadcast Technology. He is a past director of the VCE in Digital Broadcasting and Multimedia Technology and is currently Chairman and Technical Director of ProVision Communication Technologies Ltd., specialising in wireless multimedia communications. Since 2003 he has also been a member of the Science and Technology Board of the UK Defence Technology Centre in Data and Information Fusion.

David Bull has worked widely in the fields of 1 and 2-D signal processing and has published over 250 papers, various articles and 2 books. He has won two IEE Premium awards for this work. His current research is focused on the problems of image and video communications for low bit rate wireless, internet and broadcast applications. He is widely supported in these areas by both industry, Europe, MoD and EPSRC.

Nishan Canagarajah is currently a Professor of Multimedia Signal Processing at Bristol. Prior to this he was an RA and lecturer at Bristol investigating DSP aspects of mobile radio receivers. He has BA (Hons) and a PhD in DSP Techniques for Speech Enhancement both from the University of Cambridge. His research interests include image and video coding, image segmentation, content based video retrieval, 3D video and image fusion. He is widely supported in these areas by industry, EU and the EPSRC. He has been involved in a number of EU FP5 and FP6 projects where the team has been developing novel image/video processing algorithms. He has published more than 160 papers and two books. He is a member of the Exectuive Team of the IEE PN on Multimedia Communications and a member of EPSRC Peer Review College.