Reducing Processing Demands for Multi-Rate Video Encoding: Implementation and Evaluation

Håvard Espeland, University of Oslo and Simula Research Laboratory, Norway

Håkon Kvale Stensland, University of Oslo and Simula Research Laboratory, Norway

Dag Haavi Finstad, University of Oslo and Simula Research Laboratory, Norway

Pål Halvorsen, University of Oslo and Simula Research Laboratory, Norway

ABSTRACT

Segmented adaptive HTTP streaming has become the de facto standard for video delivery over the Internet for its ability to scale video quality to the available network resources. Here, each video is encoded in multiple qualities, i.e., running the expensive encoding process for each quality layer. However, these operations consume both a lot of time and resources, and in this paper, the authors propose a system for reusing redundant steps in a video encoder to improve the multi-layer encoding pipeline. The idea is to have multiple outputs for each of the target bitrates and qualities where the intermediate processing steps share and reuse the computational heavy analysis. A prototype has been implemented using the VP8 reference encoder, and their experimental results show that for both low- and high-resolution videos the proposed method can significantly reduce the processing demands and time when encoding the different quality layers.

Keywords: Multimedia, Multi-Rate, Performance, Quality Adaption, Video Encoding, VP8

INTRODUCTION

The amount of video data available in the Internet is exploding, and the number of video streaming services both live and on-demand, is quickly increasing. For example, consider the rapid deployment of public available Internet video archives providing a wide range of content like newscasts, movies and scholarly videos. In this respect, Internet users uploaded one hour of video to YouTube every second in January 2012 (YouTube, 2012). Furthermore, all major (sports) events like European soccer leagues, NFL Hockey, NBA basketball, NFL football, etc. are streamed live with only a few seconds delay, e.g., bringing the 2010 Winter Olympics (Zambelli, 2009), 2010 FIFA World Cup (Move Networks, 2008) and NFL Super Bowl (Move Networks, 2008) to millions of concurrent users over the Internet, supporting a wide range of devices ranging from mobile phones to HD displays. The number of videos streamed from such services are in the order of tens of billions

DOI: 10.4018/jmdem.2012040101

per month (Flosi, 2010; YouTube, 2012), and leading industry movers conjecture that traffic on the mobile-phone networks will also soon be dominated by video content (Cisco Systems, Inc., 2010).

Adaptive HTTP streaming is frequently used in the Internet and is currently the de facto video delivery solution. For example, Move Networks (2008) was one of the first providers of segmented adaptive HTTP streaming, and has later been followed by major actors like Microsoft (Zambelli, 2009), Apple (Pantos et al., 2010) and Adobe (2010). Recently, there are also standardization efforts by MPEG (Stockhammer, 2011). In these systems, the bitrate (and thus video quality) can be changed dynamically to match the varying bandwidth, giving a large advantage over non-adaptive systems that are frequently interrupted due to buffer underruns or data loss. The video is thus encoded in multiple bitrates matching different devices and different network conditions.

Today, H.264 is the most frequently used codec. However, an emerging alternative is the simpler VP8 which is very similar to H.264's baseline profile and supposed to be well suited for web-streaming with native support in major browsers, royalty free use and similar video quality as H.264 (Seeling et al., 2010; Ozer, 2010). Nevertheless, for both codecs, the challenge in the multi-rate scenario is that each version of the video requires a separate processing instance of the encoding software. This may be a challenge in live scenarios where all the rates must be delivered in real-time, and in the YouTube case, it will just take an enormous data center to keep the upload rate. Thus, the process of encoding videos in multiple qualities and data rates is both time and resource consuming.

Our initial idea was presented by Finstad et al. (2011) and in this paper, we expand our evaluation and further discuss our results. In particular, we analyze the processing overheads of multi-rate scenarios, and to reduce resource requirement, we investigate possibilities for reusing the output from different steps in the encoding pipeline as the same video elements are processed multiple times with only slightly different parameters. We use the VP8 processing pipeline as a case study, and we present a prototype that supports running multiple parallel VP8 encoder instances. The main contributions of our work are:

- Inspired by several transcoding approaches trying to reuse motion vectors (Kuo & Jayant, 2003; Zhou, Zhou, & Xia, 2008; Senda & Harasaki, 1999), we propose a way of reusing decisions from intra and inter prediction in a video encoder to avoid computational expensive steps that are redundant when encoding for multiple *target bitrates*, i.e., macroblock mode decision, intra prediction and inter prediction between the instances. The method can be used in any video codec comprising an analysis and encoding step with similar structure as H.264 and VP8.
- A prototype and demonstrator has been implemented using the VP8 reference encoder as a case study. We have performed a wide range of experiments using both low- and high-resolution videos with various data rates and content types. The results show that the computational demands are significantly reduced at the same rates and approximately the same qualities as the VP8 reference implementation.

RELATED WORK

Our proposed multi-rate encoder is based on the idea of sharing and reusing the results from the computational expensive steps. The idea is inspired by the Scalable Video Coding technique (Schwarz et al., 2007), but differs in key aspects: First, no special client player support is necessary. SVC has very little market penetration in comparison to VP8 player availability, which in contrast makes the multi-rate technique useful without requiring changes for the users. Second, the layers of SVC affect the output quality (Wien et al., 2007), which also may be a reason why it has not been widely adopted for online streaming. In our approach we optimize the currently used VP8 codec in a way that is more similar to transcoding. In the area of video transcoding, there are several approaches where motion vectors are reused (Kuo & Jayant, 2003; Zhou et al., 2008; Senda & Harasaki, 1999; Youn, Sun, & Lin, 1999). For example, in context of spatial downscaling, methods for synthesizing a new motion vector by reuse of the original motion vectors from the higher resolution bit-stream can be performed. In this context, Kuo and Jayant (2003) discuss transcoding with the reuse of motion vectors. Their paper investigates the statistical characteristics of the macroblocks associated with the best matching motion vectors, and they then define a likelihood score, which is used for picking the motion vectors. Similarly, Zhou et al. (2008) propose a spatial downscaling algorithm reusing the motion vectors, and a more advanced method for refining the synthesized vectors is discussed. Furthermore, Senda and Harasaki (1999) describe a real-time software transcoder with motion vector reuse. They discuss a method for reusing downscaled motion vectors, which evaluate the scaled motion vectors and their neighbors. A new method for reducing the number of candidate motion vectors is proposed, and the best one is picked by finding the one with the lowest mean absolute error. Transcoding is also investigated by Youn et al. (1999) where they observed that reusing the incoming motion vectors become non-optimal, due to the reconstruction errors. They are proposing to use a fast-search adaptive motion vector refinement to improve the motion vectors. Further, Zhou and Sun (2004) propose a new algorithm to do fast inter-mode decision and motion estimation for H.264, but both these approaches are different from our proposed solution. Fonseca et al. (2007) proposed using an open loop in H.264 that is to use the original frame as input to the prediction analysis instead of the reconstructed frame. By doing this, they showed that the quality loss was small and varied with the HD sequences evaluated. The open loop approach can be used to remove the intra-frame dependencies, and Wu et al. (2009) made a parallel version of x264 for a stream processor. Our approach does not use the original frame for prediction analysis; instead we are reusing prediction parameters for other bitrates which is closer to the reconstructed frame than the original.

We have briefly outlined some examples where the video processing reuses the output of previous executions. However, none of these papers reuse the analysis step for use with several encoder instances. In our work, we target the modern multi-rate adaptive HTTP streaming solutions with the goal of decreasing the processing requirement when uploading new content to a video archive or reducing the latency in a live streaming scenario. In summary, we investigate the possibility for reusing data from parts of the encoding pipeline to be able to output multiple video streams, and we therefore next present a brief overview of the VP8 codec. We show the basic processing pipeline, and we identify the most expensive operations by presenting some VP8 profiling results.

THE VP8 CODEC

The VP8 codec (Bankoski, Wilkins, & Xu, 2011) was originally developed by On2 Technologies as a successor to VP7, and is a modern codec for storing progressive video. After acquiring On2 Technologies in 2010, Google released VP8 as the open source *webm* project, a royalty-free alternative to H.264. The webm format was later added as a supported format in the upcoming HTML5 standard, several major browsers have implemented playback support for the format since webm is expected to be one of the major streaming formats on the web in the coming years.

Many of the features in the VP8 codec are heavily influenced by H.264. It has similar functionality as the H.264 Baseline Profile. One of the differences is that VP8 have an adaptive binary arithmetic coder instead of CAVLC. VP8 is however not designed to be an all-purpose codec; it primarily targets web and mobile application. This is why VP8 has omitted features such as interlacing, scalable coding, slices and color spaces other than 4:2:0. This reduces encoder and decoder complexity while retaining video quality for the most common use case, i.e., making VP8 a good choice for lightweight devices with limited resources.

A VP8 frame is either of type *intra-frame* or *inter-frame*, corresponding to I- and P-frames in H.264, and it has no equivalent to B-frames. In addition, VP8 introduces the concept of tagging a frame as *altref* and *golden* frames, which are stored for reference in the decoder. When predicting, blocks may use regions from the immediate previous, from the last *golden* or from the last *altref* frame. Every key frame is both a *golden* and *altref* frame; other frames can optionally be marked as *golden* or *altref*. *Altref* frames are special and never shown to the user; instead they are only used for prediction.

The encoding loop of VP8 is very similar to that of H.264. The process consists of an analysis stage, which decides if intra or inter prediction shall be used, DCT, quantization, dequantization, iDCT, followed by an in-loop deblocking filter. The result of the quantization step is entropy coded using a context adaptive boolean entropy coder and stored as the output bitstream. The output bitrate of the resulting video is dependent on the prediction parameters in the bitstream and quantization parameters.

MULTI-RATE ENCODING

The multi-rate encoder is based on the reference VP8 encoder, released as part of the webm project. Provided in Figure 1 is a simplified call graph of the VP8 reference encoder. In this call graph, we can see the flow of the program, how many times a function has been called, and how large percentage of the execution time is spent in different parts of the code. The basic flow of the entire encoder is illustrated in the upper part of Figure 2 with an analysis and encode part of the pipeline.

The *analysis* part consists of macroblock mode decision and intra/inter prediction, this corresponds to vp8_rd_pick_inter_mode in Figure 1. The *encode* part refers to transform, quantization, dequantization and inverse transform, corresponding to the functions vp8_encode_inter* and vp8_encode_intra* for the various block modes chosen. *Output* involves entropy coding and writing the output bitstream to file, this part of the encoder is not shown in the call graph. Our profiling of the VP8 encoder give similar results as for H.264 (Huang et al., 2006; Espeland, 2008), and shows that during encoding of the *foreman* test sequence, over 80% of the execution time is spent in the analysis part of the code, i.e., if this part can be reused for encoding operations for other rates, the resource consumption can be greatly reduced. This can be done since the outputs have identical characteristics except for the bitrate, and the main difference between them is the quantization parameters, that is regardless of the *target bitrate*, the analysis step which includes searching for motion vectors and other prediction parameters can be done without considering the *target bitrate*, consequently trading complexity for prediction accuracy. The prediction accuracy is reduced since the input to the analysis step does not match exactly the reconstructed frame (which has been quantized differently). The reduced prediction accuracy will lead to some degradation in quality given the same bitrate, and we will evaluate the effects in a variety of scenarios in this paper.

To evaluate this approach, we have implemented a VP8 encoder with support for multiple outputs. In our approach, we reuse a single analysis step for several instances of the encoding part as seen in Figure 2. This mitigates the requirements for re-doing the computational heavy analysis step, and at the same time allows the encoding instances to emit different output bitrates by varying the quantization parameters in the encoder step. The encoder starts one thread for each specified bitrate where each of these threads corresponds to a separate encoding instance. The instances have identical encoding parameters such as keyframe interval, subpixel accuracy, etc., except for the *target bitrate* provided. Since the bitrate varies, each instance must maintain its own state and reconstruction buffers. The threads

Figure 1. Profile of the main parts of the reference VP8 encoder

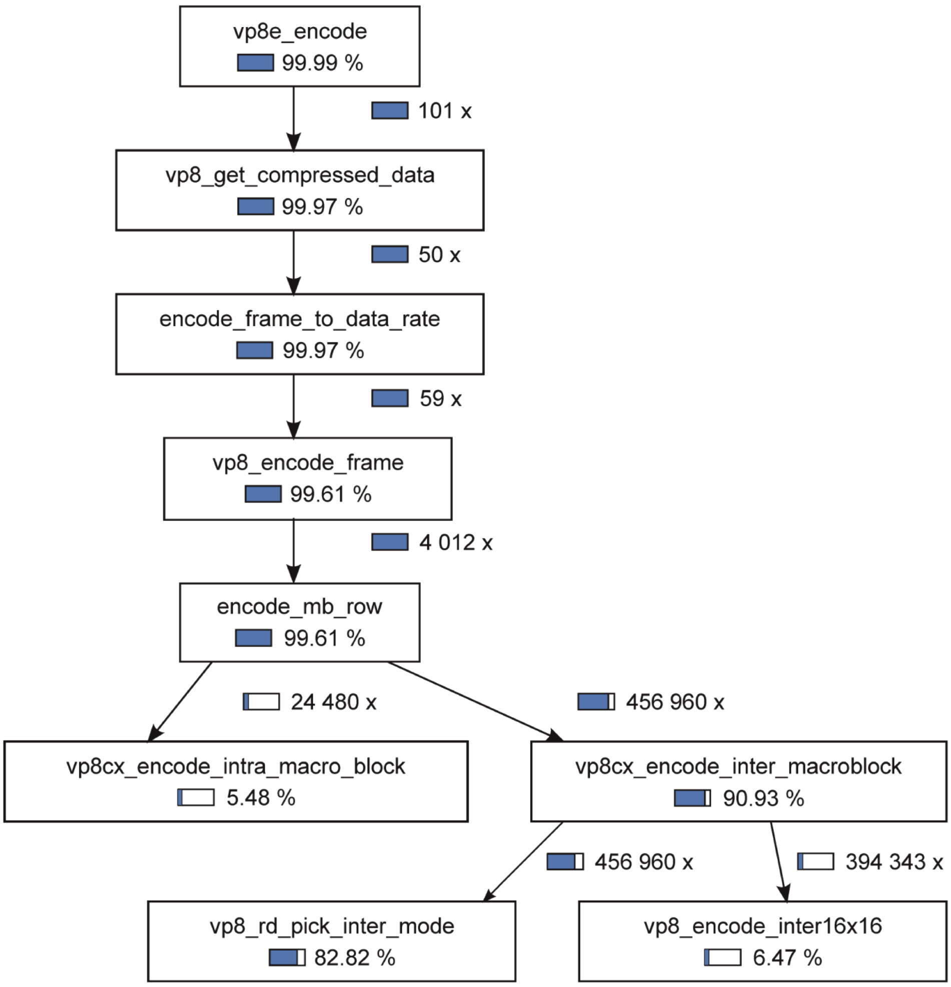
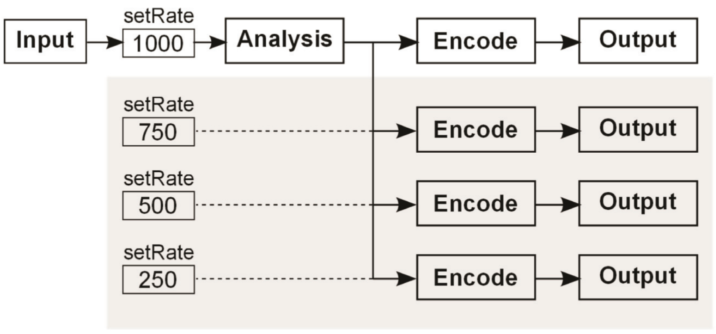


Figure 2. Encoder pipeline diagram



are synchronized on a frame by frame basis, where the main encoding instance analyses the frame before the analysis computations are made available to the other threads. This involves macroblock mode decision, intra- and inter-prediction. The non-main encoding instances reuse these computations directly without doing the computationally intensive analysis steps. Most notably vp8_rd_pick_inter_mode (Figure 1) is only performed by the main encoding instance. Since the VP8 encoder is not written with the intention of running multiple encoding instances in parallel the encoder went through significant changes in order to adapt it to run multiple instances in parallel.

EXPERIMENTS

We have performed experiments for several scenarios. One example is streaming to *mobile devices* over 3G networks. Here, Akamai (2010) recommends that video should be encoded at 250 kbps for low quality and 450 kbps for high quality. Typical 3G networks can deliver bandwidths of 384 kbps (UMTS) to 7.2 Mbps (HSDPA), and we have here measured the resource consumption for coding the standard *foreman* test sequence in *CIF* resolution using bitrates of 250, 450, 750 and 1000 kbps. Furthermore, to also test the other end of the scale, we have evaluated *HD* resolution videos. Typical ADSL lines can deliver from about 750 kbps to 8 Mbps, and for this scenario, we have performed experiments using the 1080p resolution standard test sequences *pedestrian* and *blue sky* encoded at 1500, 2000, 2500 and 3000 kbps. To measure the performance, we have used *time* to measure the consumed CPU time. Additionally, to evaluate the resulting video quality, the PSNR values are measured by the VP8 video encoder. Bitrates shown in the plots are the *resulting bitrates* achieved in output bitstreams, and not the specified *target bitrates*. *Resulting bitrates* are generally a bit lower than *target bitrates* because of a conservative rate estimator. All experiments were run on a 4-core Intel Core i5 750 processor.

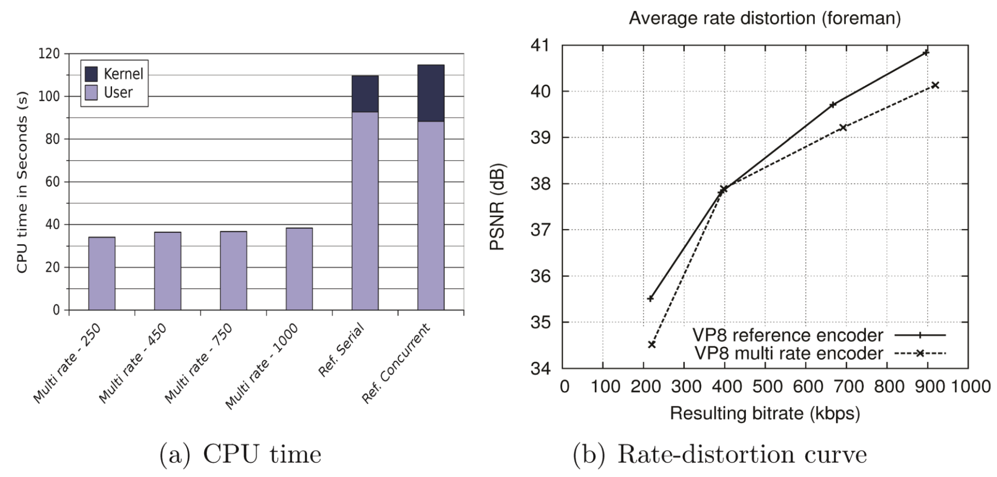
Encoding Results

To evaluate our multi-rate encoder, we have first plotted the total CPU time used when encoding the *foreman* sequence in Figure 3(a) for the four different output rates. To see if there is a difference for different chosen *prediction bitrates* when using the multi-rate encoder, we have included one test for each prediction bitrate. These results are compared to the combined CPU time used when encoding the videos for the same rates using the reference encoder with both a single thread and multiple threads. The CPU time used in the multi-rate approach is more than 2.5 times faster than encoding the four sequences using the reference encoder. The multi-rate approach scales further if the number of encoded streams is increased. In addition, the time spent in kernel space is far less in the multi-rate approach compared to the reference encoder, and we believe this is a result of reading the source video from disk only once.

To see if there are differences between low and high resolution videos, we have also looked at *HD* sequences to validate our approach. Figure 4(a) shows the *pedestrian* test clip with a *prediction bitrate* of 2000 kbps. We observe a 2.06 times reduction in CPU time for the multi-rate encoder as we saw for the *foreman* sequence.

Finally, since we reuse motion vectors for the encoding, we looked at different videos with different amount and kind of motion. The *pedestrian* has a fixed camera with objects (people) moving. The *blue sky* video has more or less fixed objects, but with a moving camera. The *blue sky* results are plotted in Figure 5(a) with a performance gain of 2.47 the performance of the reference encoder. Thus, for all our experiments using different rates, resolutions and content types, our multi-rate encoder reduce the total resource consumption.

Figure 3. CIF streaming scenario foreman



Quality Assessment

Using prediction parameters generated from a different bitrate than the *target bitrate* does have implications for the video quality. To investigate the tradeoff between reduced processing time versus degraded video quality, we have plotted a rate-distortion curve for the *foreman* sequence with a *prediction bitrate* of 450 kbps in Figure 3(b). We can see that reference encoder produces about 1 dB higher peak signal-to-noise ratio (PSNR) at the same bitrate than the multi-rate encoder. Depending on the intended usage, the significantly reduced CPU time might outweigh the small reduction in quality.

The degradation in video quality is due to the instances' reuse of analysis computations. As described in the multi-rate encoding section, the analysis part of the encoder pipeline is only carried out by the encoder instance

Figure 4. HD streaming scenario pedestrian

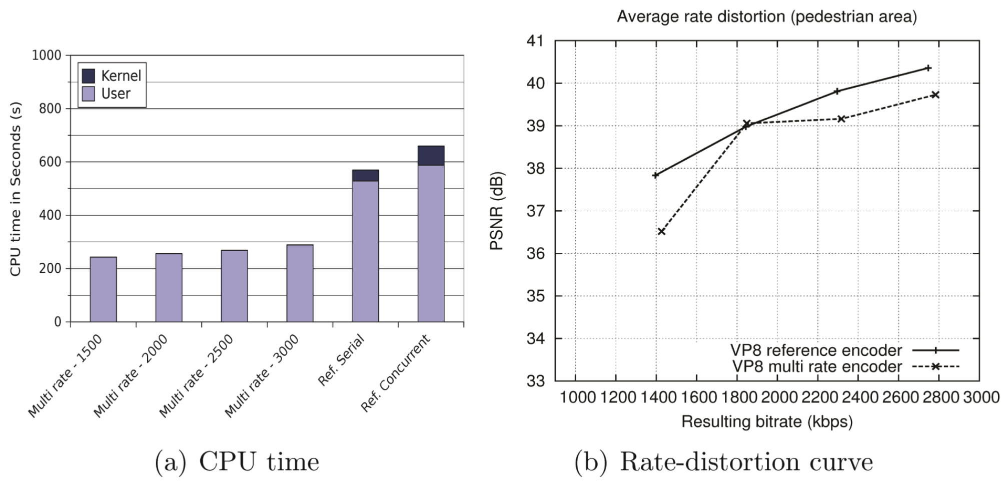
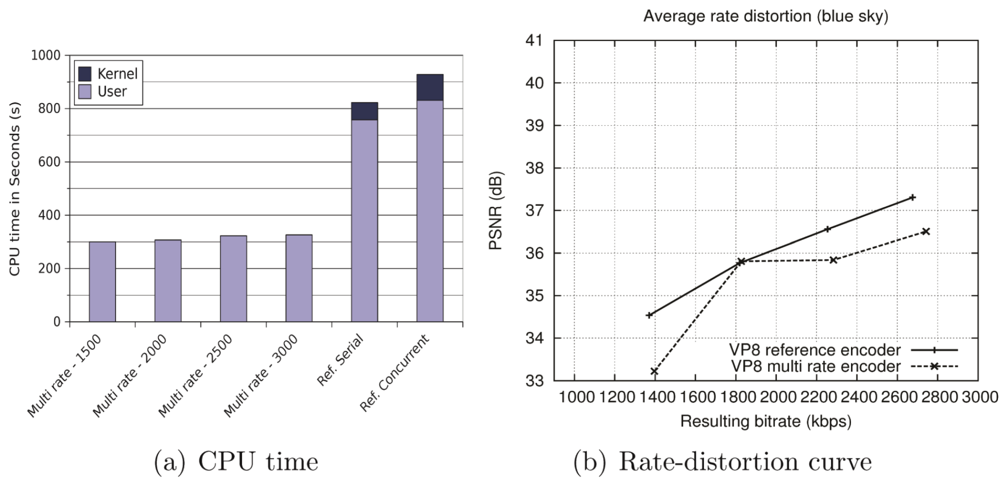


Figure 5. HD streaming scenario blue sky



(a) CPU time (b) Rate-distortion curve

targeting the *prediction bitrate*, and is hence only subject to the constraints of this instance. The mode decisions and motion vectors for the other instances will differ from the optimal parameters (as chosen by the normal version) and this leads to degradation in video quality.

Similarly, when considering the distortion of the *HD* sequences, we have plotted rate-distortion curves in Figure 4(b) and Figure 5(b) for *pedestrian* and *blue sky*, respectively. The reference encoder produces output that has 1.0 to 1.5 dB higher PSNR than the multi-rate encoder and distortion achieved for the two *HD* clips are very similar.

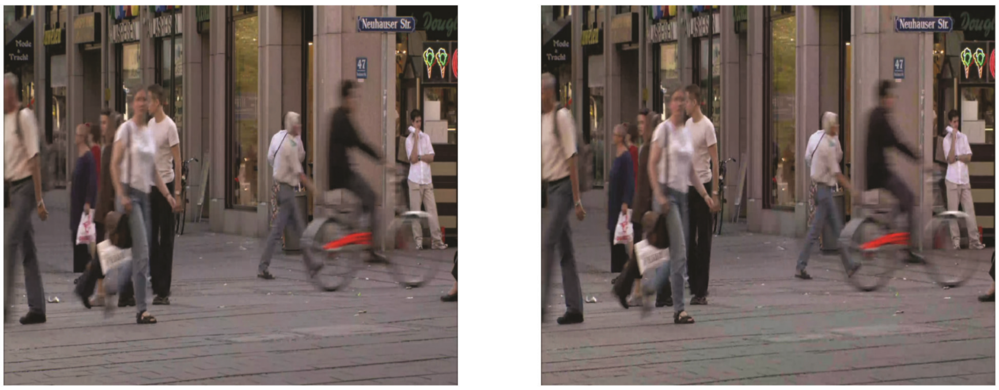
The suitability of PSNR for video quality assessment is frequently discussed, and it is often unclear what the difference means in terms of the logarithmic scale. From the plot in Figure 4(b), we can see that the PSNR of the output from the reference encoder is up to 1.32 dB better than the multi-stream encoder outputs for the *pedestrian* sequence, in the range of 1500 kbps to 3000 kbps. To see what this really means, a sample output of the "worst-case" scenario from Figure 4(b) can be seen in Figure 6. From this output, we can see that there is little visual difference between the reference encoder output and the multi-stream encoder. We also looked at the average structural similarity (SSIM) index number for the reference encoder and the multi-stream encoder. The SSIM numbers are 0.861 and 0.837, respectively, i.e., the difference is small. Thus, the quality reduction is small (we did not see a difference viewing the resulting videos, but it might be different for other types of content). In Figure 7, the "worst-case" scenario from Figure 3(b) can be seen. In this sequence, the PSNR for the reference encoder is of 0.99 dB better than the multi-rate encoder. The SSIM index numbers for this scenario are 0.873 for the reference encoder and 0.856 for the multi-stream encoder.

Choosing the Prediction Bitrate

To evaluate which *prediction bitrate* gives the minimal distortion of the videos, we have plotted rate-distortion curves for *foreman* with various prediction rates in Figure 8. We can see that the *resulting bitrate* is lower for the multi stream encoder than the reference encoder, except for when the *prediction bitrate* exactly matches the *target bitrate*, resulting in a small spike in the plot.

The lowest *prediction bitrate* (250 kbps) incurs the largest distortion difference of 2 dB for the 1000 kbps *resulting bitrate*. When using a 450 kbps *prediction bitrate*, the distortion

Figure 6. Quality difference for the "worst-case" scenario in Figure 4(b) of 1.32 dB PSNR of 1500 kbps



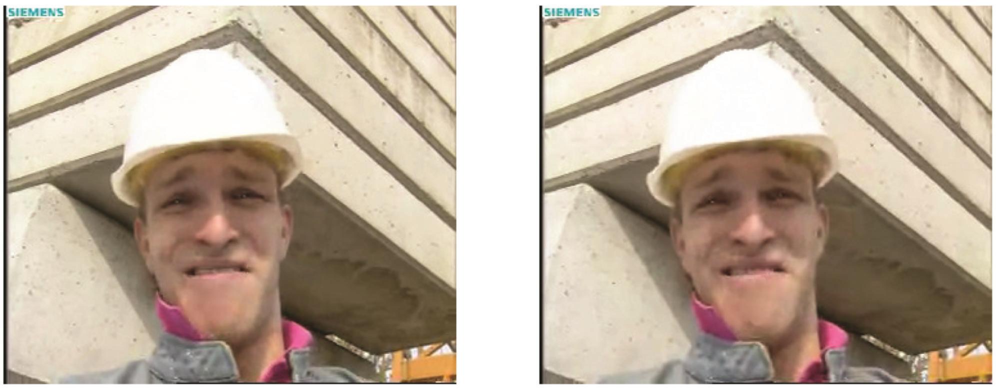
(a) Reference encoder (b) Multi-rate encoder

difference is about 1 dB for bitrates between 250 kbps and 1000 kbps. By further increasing the *prediction bitrate*, we see that the distortion difference between the multi stream and reference increases to 4 dB for the lowest output 250 kbps. Thus, the smallest distortion can be observed when using a *prediction bitrate* close to the average of the smallest and highest output bitrate, and we get a smaller penalty when the *prediction bitrate* is smaller than the output bitrate than vice versa.

Similar results can be observed when evaluating the *pedestrian* sequence, shown in Figure 9. Lower *prediction bitrates* incur less distortion difference than higher *prediction bitrates* compared to the *target bitrate*. The distortion difference is further reduced by choosing a bitrate closer to the average of the extremes.

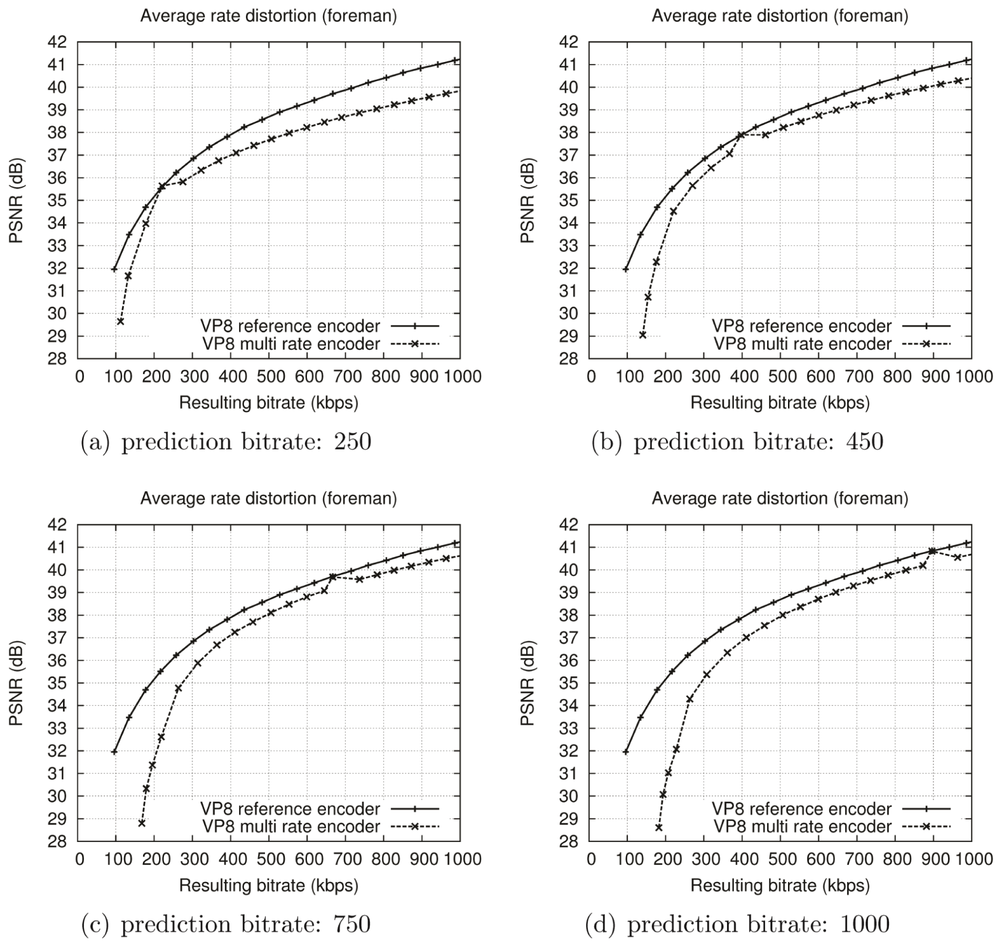
We have shown that choosing the correct *prediction bitrate* when doing multi-rate encoding has a profound effect on the quality of the output videos. Although CPU time was also

Figure 7. Quality difference for the "worst-case" scenario in Figure 3(b) of 0.99 dB PSNR of 250 kbps



(a) Reference encoder (b) Multi-rate encoder

Figure 8. Rate-distortion curve for CIF test sequence foreman with different prediction bitrate (in kbps)

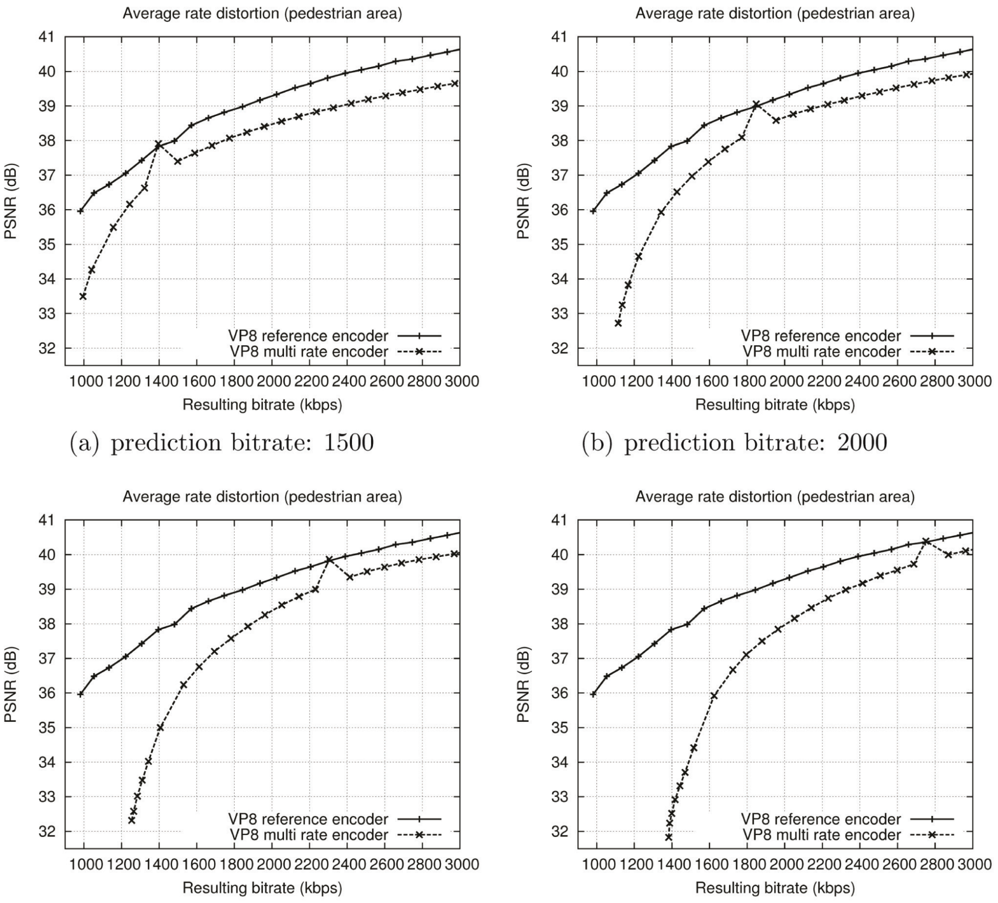


affected as shown in Figure 3(a), the difference was much less considerable. Because of the distortion, having a too wide range of *target bitrates* when doing multi-rate encoding is discouraged (see for example Figure 9(d)), but for quality ranges typically used in segmented streaming as shown in our test sequences, the results prove that multi-rate encoding is useful.

Quality Impact for Different Content Types

The previous sections have shown that our multi-output encoding approach is usable for both low- and high-resolution videos. However, encoding performance and picture quality also depend on the content type (Ni et al., 2011). In this respect, video content is classified as being high or low in both the spatial and temporal complexity (ITU-T, 1999) measured by spatial and temporal information metrics, respectively. Thus, there might be differences between videos with different amount of motion and detail as well as the resolution. To further evaluate how our multi-rate approach performs, we have performed experiments with a number of standard CIF and HD test sequences (XIPH.org, 2012) from different classifications.

Figure 9. Rate-distortion curve for HD test sequence pedestrian with different prediction bitrate (in kbps)



(a) prediction bitrate: 1500

(b) prediction bitrate: 2000

(c) prediction bitrate: 2500

(d) prediction bitrate: 3000

CIF Resolution

As we found in quality assessment section, the *foreman* sequence exhibited a PSNR quality loss compared to the reference encoder of approximately 1 dB in the range 250 - 900 Kbps. However, this difference was not visible in the example frames shown in Figure 7. Low rates gave a high loss if the prediction rate was high, but gave acceptable results for a large range of bitrates. In this section, we will investigate if this also holds for other CIF resolution videos. We used 16 standard test sequences (Table 1) using prediction rates of 100, 250, 500, 750 and 1000 Kbps. Each of these clips has different amount of detail and motion. Since we reuse motion vectors, we also include motion metrics using the MPEG 7 standard (Jeannin & Divakaran, 2001) and ITU-T-Rec-P.910 (1999) in order to see if this influences the results. We have also done tests on CPU time when encoding, and in Figure 10 we can see the speedup of our multi-rate encoder varying somewhat depending on content. However, the gain for all evaluated videos are significant compared to the reference encoder.

Table 1. CIF test sequences and their temporal complexity (higher number means more motion)

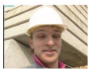
Test Sequence	MPEG7	ITU P.910	Test Sequence	MPEG7	ITU P.910
bridge-close	0.01	7.20	bridge-far	0.48	7.04
coastguard	1.57	34.91	container	0.15	9.70
flower	1.43	39.19	 foreman	7.90	36.25
hall	0.51	9.19	highway	1.51	50.79
mobile	0.37	33.02	mother-daughter	0.69	9.54
news	0.63	30.32	paris	0.17	15.89
silent	1.55	12.67	stefan	4.21	42.16
tempete	1.24	21.39	waterfall	0.00	8.24

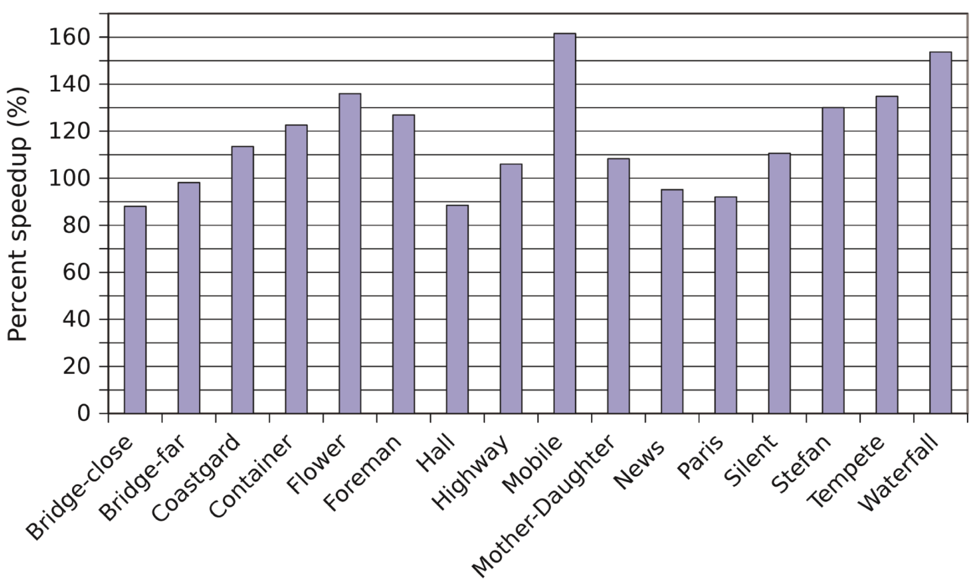
Figure 11 shows the results using 500 Kbps as prediction rate as a representative example, i.e., the tests in section indicates that a prediction rate somewhere in the middle of the target range is better. The average PSNR vary as expected between different videos, hence, the different y-axis in the plots, but we can observe that for most videos the average loss of PSNR compared to the reference encoder is less than 1 dB in the range of 250 - 900 Kbps output videos. There are exceptions, including *flower*, *mobile*, *mother-daughter*, *coastguard,* and *stefan* sequences that have higher losses compared to their references of up to 2 dB in the range. In most of these situations, there is no visible difference as also described in quality assessment section, but for some frames at the very lower end of the range, some very small visible effects can be seen. On the other hand, we can also see sequences with very little quality loss (in PSNR) such as the *paris*, *bridge-close*, *highway* and *silent* videos.

HD Resolution

For the HD resolution, we found in quality assessment section that the *pedestrian* and *blue sky* test sequences exhibited a PSNR loss of about 1.0 to 1.5 dB compared to the reference encoder. As for the CIF sequence, this difference in quality was not visible in the example frame shown in Figure 6. To see if the results are similar with other contents, we have used five HD standard test sequences available in 1080p resolution.

The results for the HD experiments are shown in Figure 12. We show only the 1500 Kbps prediction rate results as this is the best prediction rate for HD resolution according to Figure 9, but the experiments give similar results using the other rates. As in the previous section, the average PSNR is varying between the different videos as expected. However, the general trend in all the five sequences is the same: The loss of PSNR compared to the

Figure 10. Speedup at CIF resolution with Multi-Rate encoder compared to reference encoder



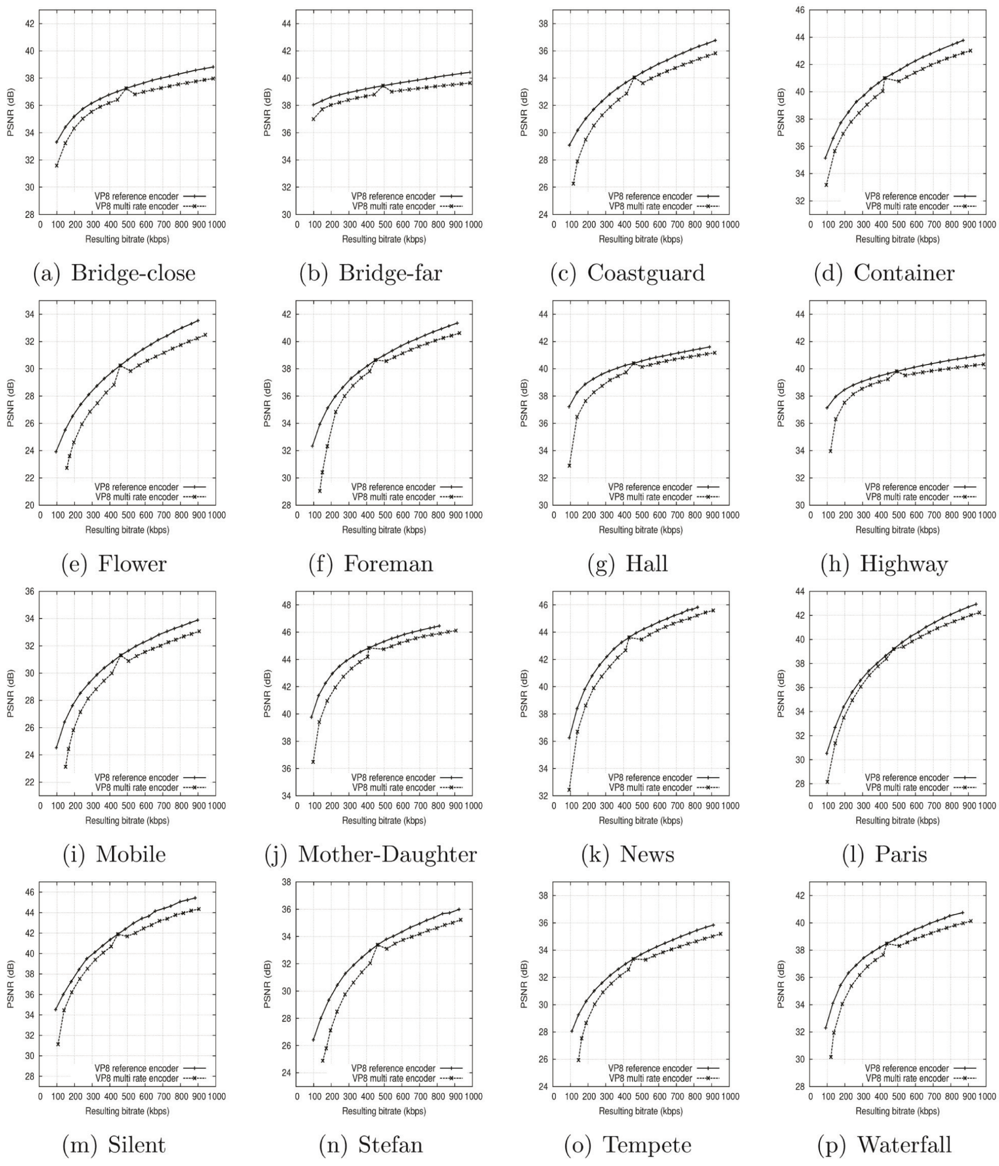
reference encoder is less than 1 dB for bitrates in the range of 1500 - 3000 Kbps. For bitrates less that 1500 Kbps, the loss of PSNR drops off faster, i.e., for all resolutions, the quality drops more and faster towards the lower end of the target interval. The worst observed difference in PSNR is slightly over 2 dB at 1000 Kbps for the *pedestrian* and *rush hour* sequence. The sequences showing the best results are the *sunflower* and *tractor* videos with worst-case PSNR less than 2 dB. As above, these differences in video quality are hardly visible to the user. We have also evaluated the CPU used for the HD sequences as can be seen in Figure 13. The results are similar to the performance of the lower resolution videos with speedup varying somewhat depending on content. Again, the gain is for all evaluated videos significant compared to the reference encoder.

DISCUSSION AND OPEN ISSUES

To prove our idea, we have implemented a prototype which reuses the most expensive operations based on a performance profile of the encoding pipeline. In particular, our multi-rate encoder reuses the analysis part consisting of macroblock mode decision and intra/inter prediction. The experimental results indicate that we can encode the different videos at the same rates with approximately the same qualities compared to the VP8 reference encoder, while reducing the encoding time significantly. However, our prototype is a small proof-of-concept implementation, and there are numerous open issues.

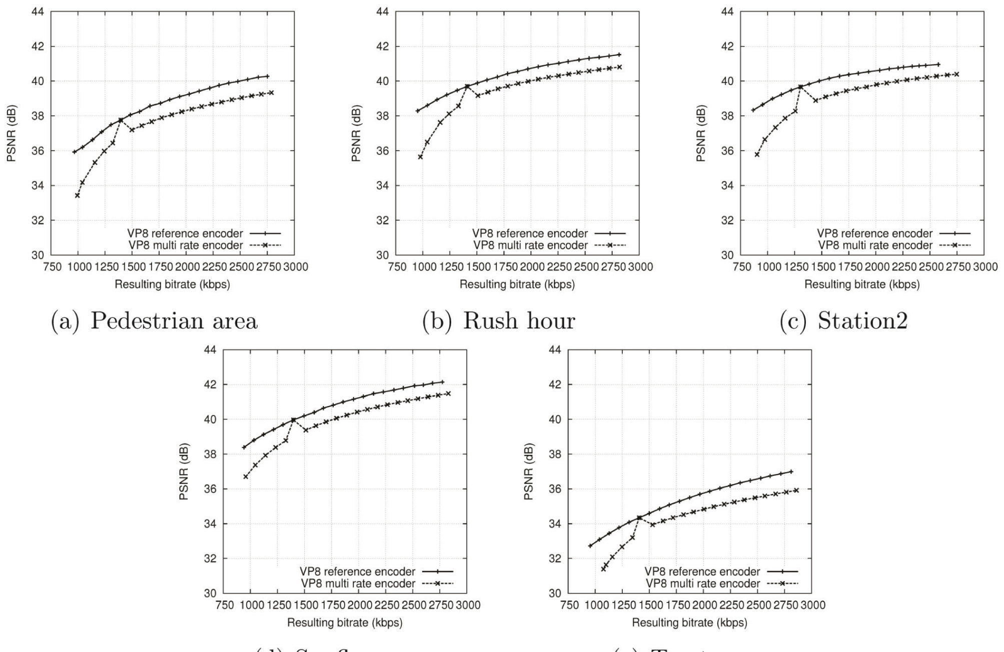
Though rarely visible, our results do show that there is a small quality degradation using our multi-rate encoder compared to the reference encoder. However, it is likely that the degradation can be reduced by further refining encoding parameters such as motion vectors to better suit the *target bitrate* from the *prediction bitrate*. One open issue is therefore to look into solutions for improving the quality for the other bitrates, aside from correctly choosing the *prediction bitrate*. By virtue of our method of reusing analysis computations directly, the quality will suffer when the *target bitrate* is not equal to the *prediction bitrate*. One plausible explanation of the quality degradation could be that videos

Figure 11. Average quality distortion for different videos using 500 kbps prediction bitrate note: different y-axis cutoff, scale retained



with high levels of motion produce many motion vectors, and that the motion vectors then chosen in the analysis stage are not meant to be used at another bitrate. To investigate this, we used two objective metrics for finding the temporal complexity of the test sequences (Table 1) and compared these metrics with the observed quality degradation. However, we cannot explain the quality degradation based on video motion alone. For example, both the *flower* and *highway* sequences feature camera panning and have similar motion complexity according to Table 1, but they result in very different quality degradations. Similarly, *stefan*

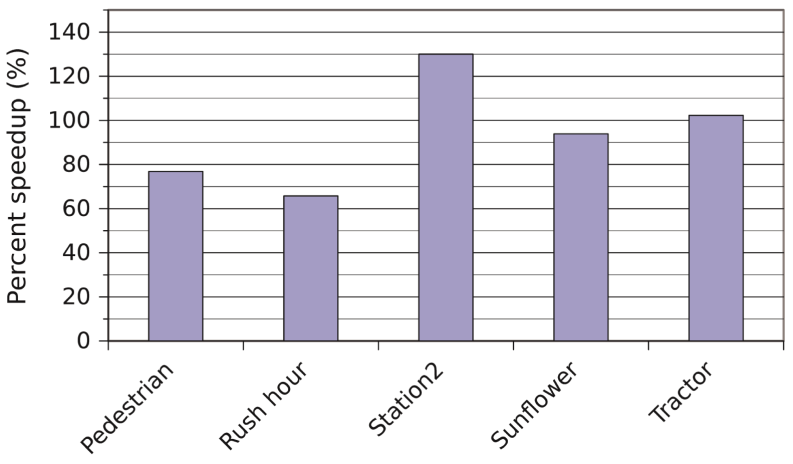
Figure 12. Average quality distortion for different 1080p videos using 1500 kbps prediction bitrate



(a) Pedestrian area (b) Rush hour (c) Station2

(d) Sunflower (e) Tractor

Figure 13. Speedup at HD resolution with Multi-Rate encoder compared to reference encoder



and *foreman* have high levels of motion, but only the former result in a quality loss of more than 1 dB. As such, we can rule out that this is caused by high levels of temporal complexity in the content alone.

Another contributor to the reduced quality compared to the regular encoder is that the input to the analysis step and the reconstructed frame are not identical (they differ in terms of quantization level). This matters since the prediction in the encoder (which uses the reconstructed frame as usual) does not produce identical pixels as the analysis step did in its prediction. The result being that motion vectors and other prediction modes are not optimally selected for the *target bitrate* since they were chosen using slightly different data. We believe this also explains the *spike* seen when the prediction rate matches the target rate, and the optimal prediction parameters are chosen; when the reconstructed frame and the frame used for prediction differ even slightly, we expect the quality to drop (since non-optimal modes can be chosen). We note that using the original frame instead of the reconstructed frame as original to the analysis stage (open loop) is a well-known technique to remove dependencies in video encoders (Fonseca et al., 2007) which reduces the prediction quality and hence the video quality. Our technique differs in that instead of using the original frame in analysis, we use the reconstructed frame for a near bitrate which is much more similar to the *target bitrate*'s reconstructed frame than the undistorted original frame.

Currently, prediction modes found in the analysis stage is used as-is without adaptations, even though it was found using another input frame (reconstructed with another quantization level). One potential quality improvement could be to do *predictor refinement*, inspired by the approach taken by Zhou et al. (2008). Since the prediction modes are found using a very similar image, we expect the motion vectors and intra-frame predictors to be close to the optimal result. Using refinements, we can improve these be doing a local search in the area near the shared prediction. This would, however, lead to increased complexity in the encoder. Moreover, the quality assessment section demonstrates how reuse of the analysis computations impacts the quality/complexity tradeoff for encoding the same input at different rates. A limitation with our multi-rate encoder is that all the bit streams encoded must use the same number of reference frames, or in the case of VP8, the same golden frames for the method to be viable. It may be possible to scale the spatial resolution in the different outputs by also scaling prediction parameters as done by some transcoding approaches, but quality impact of this is left as further work. Another potential for further work is to investigate if there are other parts of the VP8 encoder where the processing can be fanned out like in the analysis step. Also, a systematic review of all the encoder modes and decisions to pinpoint if some parameters are causing more impact than others in our prototype encoder is left as further work.

Another important point is the generality of the presented idea. In the prototype, we used VP8 as a case study since it is an emerging open-source codec, and the source code is much smaller than H.264. However, VP8 is very similar to the baseline profile in H.264, and in general, most video codecs use similar ideas for compression. Thus, our ideas are not implementation specific to VP8, but also applicable for other codecs like MPEG-1/2/4, H.263/4, VC-1, Theora, etc., which compress the video data in a similar way.

CONCLUSION

A large fraction of the Internet video services use an adaptive HTTP streaming technology. By encoding the video in multiple bitrates matching different devices and different network conditions, the bitrate (and thus video quality) can be changed dynamically to match the varying bandwidth, giving a large advantage over non-adaptive systems that are frequently interrupted due to buffer underruns or data loss. However, encoding video into multiple bitrates is a resource expensive task. We have therefore investigated the effect of running multiple encoding instances in parallel, where the different instances reuse intermediate results. This way, several encoding steps are avoided for the sub-sequent encoding operations. In particular, we have analyzed and performed experiments with Google's VP8 encoder, encoding different types of video to multiple rates for various scenarios. Our main contribution is that we propose a way of reusing decisions from intra and inter prediction in the video encoder to avoid computational expensive steps that are redundant when encoding for multiple *target bitrates* of the same video object. The method can be used in any video codec comprising an analysis and encoding step with similar structure as H.264 and VP8. Furthermore, the method has been implemented in the VP8 reference encoder as a case study, and the experimental results show that the computational demands are significantly reduced at the same rates and approximately the same qualities compared to the VP8 reference implementation, i.e., for a negligible quality loss in terms of PSNR, the processing costs can be greatly reduced. However, the quality loss is dependent on the distance from the initial bitrate, i.e., if the gap between the output bitrates is too large, the quality loss becomes larger. In such scenarios, we still need multiple instances of the whole operation.

The VP8 codec is similar H.264 and several others, and our approach should be suitable for comparable encoding pipeline as well. However, implementing a prototype and showing the same experimental results are tasks left as further work. Additionally, our aim has been to point at an operation that potentially can be optimized, and we suggested one possible solution, i.e., reusing the macro-block mode decision, intra prediction and inter prediction between the parallel encoding instances. However, as indicated in the discussion, there are several open issues where potentially other steps that can be improved with respect to the resource consumption or the resulting video quality – all promising research topics to investigate.

ACKNOWLEDGMENT

We would like to thank Pengpeng Ni for her help with evaluating the temporal complexity of test sequences. This work has been performed in the context of the *iAD* Centre for Research-based Innovation (project number 174867) funded by the Norwegian Research Council.

REFERENCES

Adobe Inc. (2010). *HTTP dynamic streaming on the Adobe Flash platform.* Retrieved from http://www.adobe.com/content/dam/Adobe/en/products/hds-dynamic-streaming/pdfs/hds_datasheet.pdf

Akamai Inc. (2010). *Akamai HD for iPhone encoding best practices.* Retrieved from http://www.akamai.com/dl/whitepapers/Akamai_HDNetwork_Encoding_BP_iPhone_iPad.pdf

Bankoski, J., Wilkins, P., & Xu, Y. (2011). *VP8 data format and decoding guide.* Retrieved from IETF website: http://tools.ietf.org/html/draft-bankoski-vp8-bitstream-01

Cisco Systems, Inc. (2010). *Visual networking index.* Retrieved from http://www.cisco.com/en/US/netsol/ns827/networking_solutions_sub_solution.html

Espeland, H. (2008). *Investigation of parallel programming on heterogeneous multiprocessors* (Master's thesis, University of Oslo). Retrieved from http://www.duo.uio.no/sok/work.html?WORKID=82232

Finstad, D. H., Stensland, H. K., Espeland, H., & Halvorsen, P. (2011). Improved multi-rate video encoding. In *Proceedings of the IEEE International Symposium on Multimedia* (pp. 293-300).

Flosi, S. L. (2010). *comScore releases April 2010 U.S. online video rankings* (Press Release). Reston, VA: comScore, Inc.

Fonseca, T. A. da, Liu, Y., & de Queiroz, R. L. (2007). Open-loop prediction in h.264/avc for high definition sequences. In *Proceedings of the Sociedade Brasileira de Telecomunicações*.

Huang, Y.-W., Hsieh, B.-Y., Chien, S.-Y., Ma, S.-Y., & Chen, L.-G. (2006). Analysis and complexity reduction of multiple reference frames motion estimation in h.264/avc. *IEEE Transactions on Circuits and Systems for Video Technology*, *16*(4), 507–522. doi:10.1109/TCSVT.2006.872783

ITU-T. (1999). *Subjective video quality assessment methods for multimedia applications* (p. 910). Geneva, Switzerland: ITU-T.

Jeannin, S., & Divakaran, A. (2001). MPEG-7 visual motion descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*, *11*(6), 720–724. doi:10.1109/76.927428

Kuo, C.-C., & Jayant, N. (2003). An adaptive non-linear motion vector resampling algorithm for down-scaling video transcoding. In *Proceedings of the International Conference on Multimedia and Expo* (pp. 229-232).

Move Networks. (2008). *Internet television: Challenges and opportunities (Tech. Rep.)*. American Fork, UT: Author.

Ni, P., Eg, R., Eichhorn, A., Griwodz, C., & Halvorsen, P. (2011). Flicker effects in adaptive video streaming to handheld devices. In *Proceedings of the 19th ACM International Conference on Multimedia* (pp. 463-472).

Ozer, J. (2010). *First look: H.264 and VP8 compared.* Retrieved from http://www.streamingmedia.com/articles/editorial/featured-articles/first-look-h.264-and-vp8-compared-67266.aspx

Pantos, R., Batson, J., Biderman, D., May, B., & Tseng, A. (2010). *HTTP live streaming.* Retrieved from http://tools.ietf.org/html/draft-pantos-http-live-streaming-04

Schwarz, H., Marpe, D., & Wiegand, T. (2007). Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology*, *17*(9), 1103–1120. doi:10.1109/TCSVT.2007.905532

Seeling, P., Fitzek, F. H. P., Ertli, G., Pulipaka, A., & Reisslein, M. (2010). Video network traffic and quality comparison of VP8 and h.264 svc. In *Proceedings of the 3rd Workshop on Mobile Video Delivery* (pp. 33-38).

Senda, Y., & Harasaki, H. (1999). A realtime software mpeg transcoder using a novel motion vector reuse and a simd optimization techniques. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. 2359-2362).

Stockhammer, T. (2011). Dynamic adaptive streaming over HTTP - standards and design principles. In *Proceedings of the 2nd ACM Conference on Multimedia Systems* (pp. 133-144).

Wien, M., Schwarz, H., & Oelbaum, T. (2007). Performance analysis of SVC. *IEEE Transactions on Circuits and Systems for Video Technology*, *17*(9), 1194–1203. doi:10.1109/TCSVT.2007.905530

Wu, N., Wen, M., Wu, W., Ren, J., Su, H., Xun, C., & Zhang, C. (2009). Streaming hd h.264 encoder on programmable processors. In *Proceedings of the 17th ACM International Conference on Multimedia* (pp. 371–380).

Xiph.org. (2012). *Test media.* Retrieved from http://media.xiph.org/video/derf/

Youn, J., Sun, M.-T., & Lin, C.-W. (1999). Motion vector refinement for high-performance transcoding. *IEEE Transactions on Multimedia*, *1*(1), 30–40. doi:10.1109/6046.748169

YouTube. (2012). *Holy nyans! 60 hours per minute and 4 billion views a day on youtube.* Retrieved from http://youtube-global.blogspot.com/2012/01/holy-nyans-60-hours-per-minute-and-4.html

Zambelli, A. (2009). *Smooth streaming technical overview.* Retrieved from http://learn.iis.net/page.aspx/626/smooth-streaming-technical-overview/

Zhou, H., Zhou, J., & Xia, X. (2008). The motion vector reuse algorithm to improve dual-stream video encoder. In *Proceedings of the 9th International Conference on Signal Processing* (pp. 2359-2362).

Zhou, Z., & Sun, M.-T. (2004). Fast macroblock inter mode decision and motion estimation for h.264/mpeg-4 avc. In *Proceedings of the International Conference on Image Processing* (pp. 789-792).

Håvard Espeland is currently a research fellow at Simula Research Laboratory and the University of Oslo at the Media Performance Group. He finished his MSc in 2008 with a thesis on parallel programming on heterogenous architectures. He stayed at UiO and started his PhD work focusing on systems support for multimedia processing. In this field he has worked on several aspects, including P2G, a framework and language for realtime multimedia processing, I/O scheduling optimizations and various architecture-level experiments and optimizations. At the University, Håvard also gives lectures on programming heterogeneous architectures. Håvard expects to submit his PhD thesis on multimedia processing by 2012.

Håkon Kvale Stensland is a research fellow at the Media Performance Group at Simula Research Laboratory and the University of Oslo, Norway. He finished his master degree (MSc) on fault tolerant routing in SCI networks in 2006. Håkon stayed at the University of Oslo and is now the lab manager at the iAD Center for research based innovation at the Department of Informatics. His research interests are within the field of distributed processing, and heterogeneous processing architectures. He is also interested in the state of the art in asymmetric multi-core processors like the nVIDIA Kepler-architecture. Håkon expects to submit his PhD thesis on the programing of multimedia workloads in heterogeneous architectures by 2012.

Dag Haavi Finstad was a master student in the Media Performance Group at Simula Research Laboratory and the University of Oslo, Norway. He received his master degree (MSc) in 2011 after implementing a prototype of the multi-rate encoder concept presented in this paper. Dag is now working as a developer at Varnish Software AS.

Pål Halvorsen is a researcher at the Media Performance Group at Simula Research Laboratory and a tenured professor in the Network and Distributed systems group at the Department of Informatics, University of Oslo, Norway. His research activities focus mostly on support for distributed multimedia systems (ranging from media on-demand systems to massive multiplayer games), including operating systems, storage and retrieval, communication and distribution. Pål received his master degree (Cand.Scient.) in 1997 and his doctoral degree (Dr.Scient.) in 2001, both from the Department of Informatics, University of Oslo, Norway.