Decoding-Workload-Aware Video Encoding

Yicheng Huang, Guangming Hong, Vu An Tran and Ye Wang
Department of Computer Science
National University of Singapore
Law Link, Singapore 117590
Republic of Singapore
{huangyic, honggm, wangye}@comp.nus.edu.sg; tranvuan82@gmail.com

ABSTRACT

This paper presents a novel decoding-workload-aware video encoding scheme. It takes raw video data and decoding workload constraint of a mobile client as input and generates a video bitstream which matches such a constraint while striving to achieve the best video quality. For a given constraint, the best overall video quality of the encoded bitstream is selected with a tradeoff between spatial and temporal distortions. The main contributions of this paper include: 1) the proposal of an efficient scheme which selects the most suitable target frame rate before the actual encoding; 2) The design of a workload control (analogous to the rate control) scheme which ensures an accurate control of the decoding workload when the bitstream is generated using the proposed encoding scheme. Experimental results demonstrate the feasibility and performance of the proposed scheme.

Categories and Subject Descriptors

I.6.3 [**Simulation and Modeling**]: Applications

H.5.1 [**Multimedia Information Systems**]: Video

General Terms

Algorithms, Design, Experimentation

Keywords

Video Encoding, Workload Control, Frame Rate Selection

1. INTRODUCTION

The recent developments in communication and microprocessor technologies have made mobile devices becoming attractive entertainment platforms for multimedia especially video applications. However, supporting video applications on mobile devices is challenging due to limited processing power on the low to middle-end mobile devices with processor frequency of 200-400 MHz. Even with the fastest mobile processor in the market today with processor frequency of around 600 MHz, it is still difficult to satisfy the decoding workload requirement of high quality video with a high frame rate.

To address this problem, we propose a decoding-workload-aware video encoding scheme to generate a video bitstream which matches the workload constraint of the target mobile devices.

The analysis in our previous work on decoding workload model [1] shows the relationship between parameters in the video bitstream and decoding workload. Given the parameters such as the number of Huffman codes, Macroblock (MB) type and motion vectors' precision, the decoding workload can be estimated accurately using the proposed decoding workload model. This inspires us to design a decoding-workload-aware video encoding scheme, which controls the decoding workload from the encoder side by adjusting these parameters during the encoding process. The concept of workload control is analogous to the rate control which controls the target bitrate to satisfy the bandwidth/storage constraint. The proposed workload control scheme is implemented with an MPEG-2 video codec for the proof of concept.

For video encoding, the frame rate plays a significant role on the decoding workload. Intuitively, a video bitstream with the same image quality but a lower frame rate has a lower decoding workload. For a conventional video encoder, the target frame rate is typically fixed at 25 or 30 fps. However, mobile devices with low processing power might not be able to decode a video bitstream with such a high frame rate in real time. To reduce the decoding workload, we can either reduce the frame rate or reduce the quality of the individual frame. The problem is, given a decoding workload constraint, there can be more than one candidate with different combinations of frame rate and individual frame quality. The challenge is how to choose a candidate with the best overall quality. Conventional objective video quality measures such as MSE and PSNR cannot compare quality between video clips with different frame rates. A possible solution proposed in [2, 3] is to replace the dropped frame by its previous frame in display order and calculate average PSNR or MSE of the new video sequence as the video quality. The rationale behind such approach is that a video player typically maintains the current frame on the screen before displaying the next frame. With this method, both spatial distortion (image quality) and temporal distortion (frame rate) are considered. This approach, however, is problematic for our proposed encoding scheme, because it requires encoding, decoding and calculating PSNR/MSE for every candidate of all possible frame rates. This could become prohibitively expensive computationally. To solve this problem, we propose a new frame rate selection scheme which provides a fast estimation of the resulted distortion for every candidate before the actual encoding. This is followed by a workload control scheme which accurately

NOSSDAV'08, May 28–30, 2008, Braunschweig, Germany.

determines the decoding workload of the generated video bitstream when being decoded on the targeted clients. The architecture of the proposed decoding-workload-aware encoder is shown in *Figure 1*.

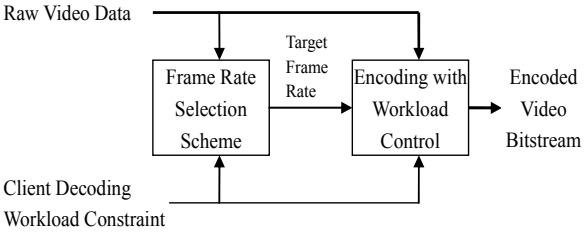

Figure 1: *The encoder architecture*

The encoding procedure includes two phases. In the first phase, the raw video data and client decoding workload constraint together with all possible frame rates are fed into the *Frame Rate Selection Scheme* which selects the most suitable frame rate for the actual encoding. In the second phase, the encoder uses the selected frame rate and client decoding workload constraint to compress the raw video into the target video bitstream using the *Workload Control Scheme*.

The rest of the paper is organized as follows: Section 2 introduces the workload control scheme. Section 3 presents the frame rate selection scheme. Section 4 includes evaluations of the proposed schemes and Section 5 concludes this paper.

2. WORKLOAD CONTROL SCHEME

According to the decoding workload model proposed in [1]: an MPEG video is made up of a sequence of Macroblocks (MBs), and modeling the video decoding workload is decomposed to modeling three major tasks of decoding one MB: Variable Length Decoding (VLD), Inversed Discrete Cosine Transform (IDCT) and Motion Compensation (MC). The decoding workload of VLD is modeled as a linear function of the number of Huffman codes. The decoding workload of IDCT is modeled as a lookup table indexed by the last position of Huffman codes. The workload of MC is modeled as a lookup table indexed by motion vectors' precisions. And for different types of MB, the parameters of the models can be different. Thus, we can control the decoding workload by adjusting the number of Huffman codes, MB type and motion vector precision.

The workload control can work at three levels: frame level, MB level or task level. For an encoder, workload control in frame and MB level is similar to the conventional bit rate control. However, in task level, rate control and workload control are significantly different. Rate control scheme only considers the quantization level of DCT coefficients, which is proportional to the bit rate. Workload control needs to consider multiple factors and their tradeoff. For example, if we allocate more workload to the VLD or IDCT task, we can have more Huffman codes. This increases the video quality. However, allocating more workload to the VLD or IDCT task will result in allocating less workload to the MC task. This may limit the motion vectors' precision and thus decreasing the video quality. This problem becomes even harder if we also consider the MB type. In this section, as the first step of our research on the decoding-workload-aware encoder, we do not consider the task level workload control: we simply fix the workload of the MC task by fixing the MB type and motion vectors from the conventional motion estimation procedure. We only control the decoding workload of IDCT and VLD tasks by adjusting the number of Huffman codes.

We design two strategies for the frame level and MB level workload control, respectively. The strategies allocate the workload so that the encoded bitstream can have a better video quality within the constraint of decoding workload. The two strategies can be summarized as follows:

- In frame level, the workload is allocated based on statistical ratio of different frame types and the ratio is adjusted according to the recent history.
- In MB level, the workload is allocated based on the image complexity which can be estimated by the variance or MSE.

The experiment results in Section 4 show that these two strategies can improve the video quality. *Algorithm 1* describes how the workload control scheme works.

1) Allocate the workload for the current GOP according to the constraint, GOP size and frame rate on an average basis.

2) For all the frames in the GOP:

3) Allocate the workload to the current frame according to the frame type and history record.

4) Run motion estimation for all the MBs of the current frame, decide their MB types, record their MSEs (or VAR for I-MB) and motion vectors.

5) Estimate the workload of MC for all the MBs based on the results from 4) using the workload model.

6) For all the MBs in the current frame:

7) Allocate the workload of current MB by its MSE/VAR and MB type.

8) From the motion vectors get in the line 4, estimate the workload of VLD+IDCT for all possible number of quantization scales using workload model. Select out the number of quantization scale having the workload closest to $W_{VLD\_IDCT} = W_{mb} - W_{MC}$, where W_{mb} is the workload allocated for the MB, and W_{MC} is the workload of MC get in line 5.

9) Encode the MB.

10) Update the status.

Algorithm 1: *Workload Control Scheme*

The details are as follows:

In line 3, the decoding workload for current frame (W_i, W_p and W_b for I-frame, P-frame and B-frame) is allocated as:

$$W_i = W / (1 + \frac{N_p X_p}{X_i K_p} + \frac{N_b X_b}{X_i K_b}) \quad (1)$$

$$W_p = W / (N_p + \frac{N_b K_p X_b}{X_p K_b}) \quad (2)$$

$$W_b = W (N_b + \frac{N_p K_b X_p}{X_b K_p}) \quad (3)$$

where X_i, X_p and X_b are the decoding workload for the previous I-, P- and B-frames; K_p and K_b are the parameters representing the

ratio between I-, P- and B-frame. In our implementation, $K_p=1.0$ and $K_b=2.0$, which are obtained empirically. W is the remaining workload of the GOP, which is updated after encoding a frame. N_p, N_b are the number of P-and B-frames in a GOP. At the beginning, X_i, X_p and X_b are initialized as

$$X_i = W \Big/ \left(1 + \frac{N_p}{K_p} + \frac{N_b}{K_b}\right)$$

$$X_p = W \Big/ \left(N_p + \frac{N_b K_p}{K_b}\right)$$

$$X_b = W \Big/ \left(N_b + \frac{N_p K_b}{K_p}\right)$$

In line 4, we use conventional motion estimation, with which the MB type is decided by comparing the MSE or VAR of the MB with a constant threshold.

In line 7, workload of the current MB, $W_{mb}(i)$ is allocated as

$$W_{mb}(i) = (W_{frame} - \sum_{j=i}^{N} W_{MC}(j)) * MSE(i) / \sum_{j=i}^{N} MSE(j) + W_{MC}(i) \quad (4)$$

where W_{frame} is the remaining workload of the current frame; $MSE(i)$ is the MSE of the i^{th} MB (or $VAR(i)$, if the i^{th} MB is an I-MB); N is the number of MBs of the frame. $W_{MC}(i)$ is the workload of MC of the i^{th} MB. The rationales behind this equation are: 1) as mentioned earlier, we do not change motion vectors' precision or MB type after motion estimation, the workload of MC can be regarded as fixed; 2) we allocate more workload to VLD and IDCT tasks of the MB which has larger MSE/VAR. A MB with larger MSE/VAR implies more residual error. Therefore, it requires more Huffman codes for the encoding. And to decode a MB with more Huffman codes, more decoding workload is required in VLD and IDCT tasks.

In line 9, we use the quantization scale obtained from line 8 for the actual encoding (generating the bitstream). If the encoder also employs a rate control scheme, we will get another quantization scale for the rate control. In this case, both the decoding workload constraint and bit rate constraint can be satisfied by selecting the larger quantization scale.

In line 10, we estimate the decoding workload using the parameters extracted from the encoded MB and update status of the scheme.

To summarize, in the frame level, we allocate the workload based on statistical ratio between different components which is updated with a moving average of recent history; in the MB level, we allocate the workload based on the image complexity. The experiment results in Section 4 show that these two strategies improve the video quality considerably.

3. FRAME RATE SELECTION SCHEME

This section presents our fast frame rate selection scheme: it enumerates all the frame rate candidates; for each candidate, the distortion of the target video bitstream is estimated. The candidate with the smallest distortion is selected. The problem is how to perform a fast distortion estimation of all target video bitstreams with all possible frame rates before actual encoding.

Before going to the detail of the algorithm, we introduce some notations first. Assume we have a raw video sequence containing N frames: $P(0), P(1), P(2)…P(N-1)$ (see *Figure 2*). For each frame rate candidate f, we evenly select $M=N*f/f_{max}$ frames from the original sequence for actual encoding, where f_{max} is the maximum frame rate. In our implementation, f_{max} is set as 25fps. We denote $P'(0),P'(1),P'(2),…,P'(M-1)$ are the frames decoded at the client end. In *Figure 2*, f is equal to 12 fps. Replacing a dropped frame by its previous frame, we get the frame sequence $P'(0,0),P'(0,1)..P'(0,f_{max}/f-1),P'(1,0),P'(1,1)…P'(1,fmax/f-1)…P'(M-1,0),P'(M-1,1)…P'(M-1, f_{max}/f-1)$, where $P'(i,j)$ is exactly the same as $P'(i,0)$. And $P'(i,j)$ is corresponding to the frame $P(i*f_{max}/f+j)$ in the original video sequence.

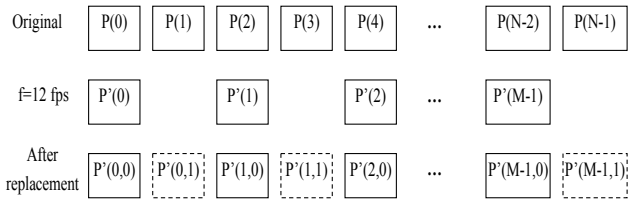

Figure 2: *An example for illustrating frame rate selection scheme*

The distortion of the video sequence with frame rate f is calculated by the distortion of the video sequence after frame replacement, which is then calculated by the average distortion of the corresponding frames. Here are the basic ideas to calculate the distortion between two corresponding frames:

For the frames *P'(i,0)*

According to the frame rate and GOP structure, we know the frame type of $P'(i,0)$. Using a simple version of workload control scheme in Section 2, we can also estimate the number of Huffman codes in this frame.

If P'(i,0)is an I-frame,

The variance of the frame describes its image complexity. The distortion between $P'(i,0)$ and $P(i*f_{max}/f)$ is estimated as the image complexity lost when being encoded into the target bitstream due to workload constraint, i.e., a MB will be coded using just a part of the total 64 Huffman codes:

$$D(i,0) = Var(i * f_{\max} / f) * (1 - \frac{w_{huff}(N)}{w_{huff}(64)}) \quad (5)$$

where N is the number of Huffman codes, $W_{huff}(N)$ is the weight of the first N Huffman codes, $Var(i)$ is the variance of the original frame, which can be calculated before the actual encoding.

If P'(i,0) is a P-frame,

A P-frame is dependent on its reference frame. Assuming the reference frame is $P'(k)$, its distortions have two parts: the distortion propagated from its reference frame and the residual error lost due to the workload constraint:

$$D(i,0) = W_{prop} * D(k,0) + \mathrm{Res}'(k,i) * (1 - \frac{w_{huff}(N)}{w_{huff}(64)}) \quad (6)$$

where W_{prop} is the weight representing the error propagation effect, *D(k, 0)* is the distortion of *P'(k)* which can be calculated by Eq. 5 (if *P'(k)* is an I-frame) or Eq.6 (if *P'(k)* is a P-frame). *Res'(k,i)* is the residual error between *P'(k)* and *P'(i)*. The residual error is calculated in the motion compensation procedure. Since running the motion compensation for all frame rate candidates is computationally expensive, we estimate *Res'(k,i)* by

$$Res'(k,i) = Res(p,q) = Res(p,p+1) + W_{res} * (Res(p+1,q)) \quad (7)$$

where $p=k* f_{max}/f$, $q=i* f_{max}/f$, W_{res} is a parameter. Thus, we can estimate the residual error between any two frames by a linear combination of the residual error between two adjacent original frames, which needs to be calculated only once before the actual encoding.

If P'(i,0) is a B-frame

It depends on two frames *P'(k)* and *P'(t)*. Similar to the P-frame, its distortion can be calculated as:

$$D(i,0) = W_{prop} * \frac{D(k,0)+D(t,0)}{2} + \frac{Res'(k,i)+Res'(i,t)}{2} * (1 - \frac{w_{huff}(N)}{w_{huff}(64)}) \quad (8)$$

For the frames P'(i,j), where j>0

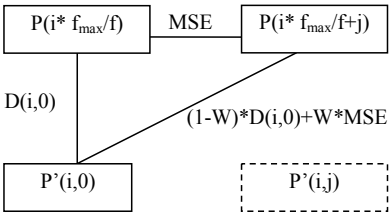

Figure 3:*The distortion calculation for P'(i,j)*

The distortion between *P'(i,j)* and its corresponding frame *P(i* $f_{max}/f+j$)* equals to the distortion between *P'(i,0)* and *P(i*$f_{max}/f+j$)*, since *P'(i,j) is a direct copy of P'(i,0)*. Using the frame *P(i*f_{max}/f)* as a bridge (see *Figure 3*), the distortion can be estimated as a weighted sum of the spatial distortion between *P'(i,0)* and *P(i* f_{max} */f)*, and the temporal distortion between *P(i* $f_{max}/f+j$)* and *P(i* f_{max} */f)*.

$$D(i,j) = (1-W_{temp}) * D(i,0) + W_{temp} * MSE(i*f_{max}/f,\ i*f_{max}/f+j) \quad (9)$$

where W_{temp} is the weight of the temporal distortion and *MSE(i*fmax/f, i*fmax/f+j)* is the MSE between *P'(i*fmax/f) and P(i*fmax/f+j)*, which is used to represent the temporal distortion caused by the frame replacement in the display sequence. Again, we do not want to calculate MSE for all the possible candidates. Let $p=i*f_{max}/f$ and $q=i*f_{max}/f+j$, we estimate *MSE(p,q)* by

$$MSE(p,q) = MSE(p,p+1) + W_{mse} * MSE(p+1,q) \quad (10)$$

Thus, we estimate the MSE between any two frames by a linear combination of the MSE between two adjacent original frames, which needs to be calculated only once before the actual encoding.

The detail of the distortion estimation for each frame rate candidate is shown as *Algorithm 2*.

1) Select the frames, *P'(i,0),i=1…M,* from original sequence based on the frame rate.

2) For all the selected frames:

3) Allocate the workload to the current frame according to the frame type.

4) Estimate the sum of the workload of MC, W_{MC} of all the MBs in the current frame

5) Estimate the sum of the workload of IDCT+VLD, $W_{VLD\_IDCT}=W- W_{MC}$, where *W* is the workload of the frame.

6) Estimate the average number of Huffman coefficients of MB in the frame.

7) Estimate the distortion *D(i,0)* between *P'(i,0)* and *P(i* f_{max}/f*)*.

8) Replace the dropped frame by its previous frame. For all the replaced frames, *P'(i, j),i=1…M,* estimate the distortion *D(i,j)* between *P'(i,j)* and *P(i*$f_{max}/f+j$*), j=1..* f_{max}/f.

9) Calculate *Avg(D(i,j))* as the overall distortion of the target video sequence.

Algorithm 2: *Frame Rate Selection Scheme*

The details are as follows:

In Line 3, frames with different type are allocated using different ratio. We keep the ratio the same as that in *Algorithm 1*: 2:2:1 for I-, P- and B-frame.

In Line 4, we assume all MBs in I-frame are I-MBs and 1/3 MBs in P-frame are I-MBs and another 2/3 MBs are P-MBs. We also assume 1/2 MBs in B-frame are P-MB and another 1/2 MBs are B-MBs. Based on this ratio, we estimate the sum of the workload of MC of the MBs in the current frame. It should be noted that above approach is not the most accurate one. A more accurate approach can employ the residual error to estimate the number of I-, P- and B-MBs of the frame. However, our simple scheme is designed to select the best frame rate. Experimental results show that this simple approach works sufficiently well. For I-, P- and B-MB, we use a constant value to estimate the MC workload. The constant value is obtained from statistical analysis. Again, this is not the most accurate approach, but is sufficient for our purpose.

In Line 6, the number of Huffman codes is estimated using the decoding workload model in [1].

In Line 7, the distortion *D(i,0)* is calculated as *Eq 5, 6* and *8*.

In Line 8, the distortion *D(i,j)* is calculated as *Eq 9*.

In the proposed scheme we have many parameters such as $W_{huff}(N)$, W_{temp}, W_{prop}, W_{mse} and W_{res}. They are all obtained from the statistical analysis: $W_{huff}(N)$ is obtained from the experiment where we select a 8*8 block from a raw picture, performing the DCT operation, setting the coefficients after position N as zero and finally calculating the difference between the original block and the block after IDCT. For W_{temp}, W_{prop}, W_{mse} and W_{res}, we use a set of video as the training set. We enumerate the four parameters from 0~10 with a step of 0.1 and select the values with best estimation result.

4. EVALUATION

4.1 Workload Control Scheme Evaluation

4.2.1 Experimental Setup

For proof of concept of the proposed decoding-workload-aware video encoding, we employ the MPEG-2 as the video format. We modify MPEG-2 reference encoder to a decoding-workload-aware encoder. In our experiments, we select 12 raw video sequences which are shown in *Table 1*. Each of them is encoded under 11 workload constraints: 20 MHz, 30 MHz, 40 MHz, 50 MHz, 60 MHz, 80 MHz, 100 MHz, 120 MHz, 150 MHz and 200 MHz. We use MPEG-2 decoder of TCPMP project [4] as the target decoder. We use SimpleScalar [5] to simulate the decoding procedure and record the actual decoding workload, which is then compared with the workload constraints.

No	Video Name	Description
1	akiyo	Still background and a foreground object with very low movements.
2	bridgeclose	Still background and some small objects with random movements.
3	bridgefar	Almost a still image.
4	coastguard	Still background and two foreground objects with contrary movements.
5	container	Still background and two foreground objects with same movements.
6	foreman	Background and foreground have moderate movements.
7	hall	Still background and two objects with moderate movements.
8	highway	Background with very fast movements.
9	mother-daughter	Still Background and two objects with very slow movements.
10	news	Still background, an object with fast movements and two objects with very low movements.
11	silent	Still background and an object with moderate movements
12	walk	Both background and two foreground objects are with very fast movements

Table 1: *12 raw video sequences*

4.2.1 Experimental Results

Figure 4 shows the comparison between the constraint and the actual decoding workload for the sequence 'akiyo'. We run 10 experiments, where we set the workload constraint as 20, 30, 40, 50, 60, 80, 100, 120, 150 and 200MHz, respectively. The results of the other sequences also show similar matches. Two curves in the figures represent the constraint and the actual decoding workload, respectively. It can be observed that, in most cases, the actual workload is very close to the constraint. However, when the constraint is very low (20MHz), the actual decoding workload is more than the constraint as seen in Figure 4. It is because each sequence has a minimum decoding workload requirement which depends on the video content. For example, when the motion of the video is large, the residual error in P- and B-frame is large as well, which demands more decoding workload. Thus the minimum decoding workload requirement will be large, and vice visa. The average difference between the constraint and actual decoding workload is less than 1.8 %. This indicates that the workload control scheme controls the decoding workload very well.

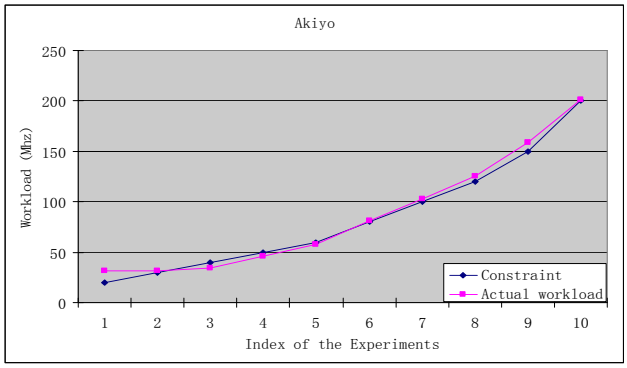

Figure 4: *The comparison between the constraint and actual decoding workload for sequence 'akiyo'.*

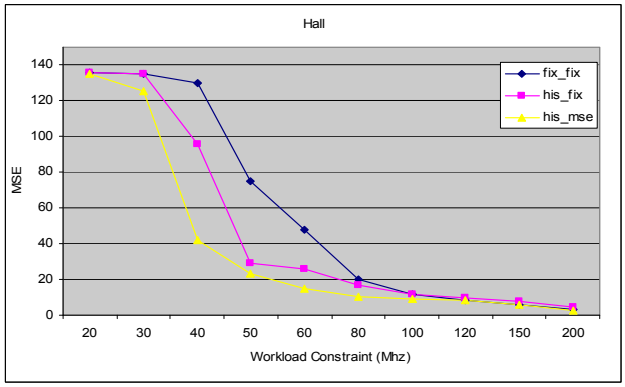

Figure 5: *The comparison between the video distortions between different workload control schemes for the sequence 'hall.*

Next, we evaluate the strategies we employed in the workload control scheme. In *Figure 5* we compare the video quality of the bitstream generated by our scheme with the bitstreams generated without employing the strategies for the video sequence 'hall'. The x-axis represents the workload constraint and the y-axis represents the MSE of the encoded video bitstream. The results of the other video sequences show similar patterns. In *Figure 5*, the curve *his_mse* represents MSE value of the bitstream generated by the scheme using both strategies described earlier. The curve *his_fix* represents the MSE value of the bitstream generated by the scheme only using the strategy in frame level; in the MB level, we allocate workload according to a fixed ratio. And the curve *fix_fix* represents the MSE value of the bitstream generated without using any strategy, i.e., we allocate workload based on fixed ratios in both frame and MB level. It is observed that, under the same constraint, the bitstream generated by using both strategies has better quality than the bitstream generated by using only one strategy on the frame level, which is still better than the bitstream generated without using any strategy. Our experimental results show that both workload allocation strategies in the frame and MB levels are effective.

4.2 Frame Rate Selection Scheme Evaluation

4.2.1 Experimental Setup

In the experiment, given a raw video sequence and the workload constraint, we select the best frame rate from the candidates using the frame rate selection scheme. To evaluate the result, we encode

and decode the sequence under the same workload constraint for all the frame rate candidates. Then, the dropped frames are replaced with the previous un-dropped frames and the average MSE is calculated. We evaluate if the frame rate selected by our scheme has the smallest MSE. In the experiment, we use 12 different video sequences and 14 workload constraints: 10, 20, 30, 40, 50, 60, 80, 100, 120, 150, 180, 200, 250 and 300 MHz; and frame rate candidates are 5, 10, 15, 20 and 25 fps.

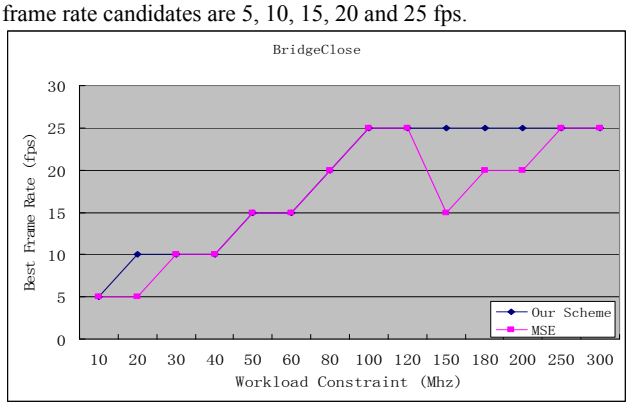

Figure 6: *The comparison between our scheme and MSE for the sequence 'bridgeclose'*

4.2.2 Experiment Results

Figure 6 shows the comparison between our scheme and MSE for the sequence 'bridgeclose'. The results for the other sequences show similar patterns. It is observed that our scheme and MSE match well in most cases. The percentage that the frame rate selected by our scheme has the smallest MSE value is 74.4%. The percentage that the frame rate selected by our scheme has the smallest or second smallest MSE value is 90.4%. Furthermore, the cases our scheme does not match the MSE, are possibly because MSE does not reflect the video quality accurately. For example in *Figure 6*, when the workload constraint is 100 and 120 MHz, MSE selects the best frame as 25fps. However, when the workload constraint increases to 150, MSE selects the best frame as 15fps which is counter-intuitive. The best frame rate should not decrease with the increase of workload constraint. The case shown in Figure 6 illustrates that our scheme is more reliable than the conventional MSE.

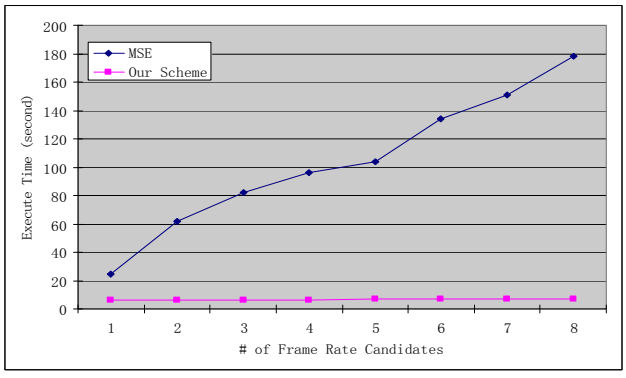

Figure 7: *The complexity comparison between the two schemes*

In comparison with the conventional approach, such as MSE, our scheme has a much lower computational complexity. If we use the conventional approach, we have to encode, decode and calculate MSE for *n* times, where *n* is the number of the frame rate candidates; while in our scheme, we run the motion estimation (a part of the encoding process), calculate MSE and variance only once. A comparison of time complexity of the two schemes is shown in Figure 7. The test was run on a desktop with Pentium 4 CPU and 1G RAM running Windows XP. As shown in Figure 7, the execution time increases with the number of frame rate candidates for the conventional approach, while the execution time for the proposed scheme is almost constant. When the number of frame rate candidates is 8, our scheme is about 25 times faster than the conventional approach.

5. CONCLUSION

In this paper, we have presented a new decoding-workload-aware video encoding scheme with two main contributions: a decoding workload control scheme and a fast frame rate selection scheme. The workload control scheme can control the decoding workload accurately when the generated video bitstream using the proposed scheme is decoded in a target client. The fast frame rate selection scheme can select out the most suitable target frame rate, balancing the spatial and temporal distortions, before the actual encoding.

We believe that the proposed fast frame rate selection scheme is not only useful for workload control but also for rate control. On the other hand, our workload control scheme still has a lot of room for improvement. For example, the workload allocation in the task level is an important and interesting problem to study in the future.

Another exciting future work lies on the relationship between the decoding workload and processor energy consumption. We can use the same principle presented in this paper to control the energy consumption level of the video decoding devices via dynamic voltage/frequency scaling for the purpose of extending battery life or simply matching the current battery level.

6. REFERENCES

[1] Y. Huang, V. Tran, Y. Wang, "A Workload Predication Model for Decoding MPEG Video and its Application to Workload-scalable Transcoding", ACM Multimedia Conference, pp. 952-961, September, 2007.

[2] M. Bonuccelli, F. Lonetti, F. Martelli, "Temporal Transcoding for Mobile Video Communication", the second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 18- 29, March, 2005.

[3] K. Ngan, T. Meier, Z. Cheng, "Improved Single-video Object Rate Control for MPEG-4", Circuits and Systems for Video Technology, IEEE Transactions on, pp. 385-393, May, 2003.

[4] http://tcpmp.corecodec.org.

[5] T. Austin, E. Larson, D. Ernst, "Simplescalar:An infrastructure for computer system modeling", IEEE Computer, pp. 59-67, 2002.