Latency Analysis for a Multi-Processor Multiview Video Encoder Implementation

Pablo Carballeira,* Julián Cabrera,* Antonio Ortega,† Fernando Jaureguizar,* and Narciso García*

* Grupo de Tratamiento de Imágenes, ETSI de Telecomunicación, Universidad Politécnica de Madrid, Spain
E-mail: {pcl, jcq, fjn, ngs}@gti.ssr.upm.es
† Signal and Image Processing Institute, Department of Electrical Engineering, University of Southern California,
Los Angeles, CA 90089-2564, USA, E-mail: antonio.ortega@sipi.usc.edu

***Abstract*—In this paper we extend a previous encoding latency analysis framework for arbitrary multiview video coding prediction structures with a new multi-processor multiview encoder model. As opposed to the fixed scheme where each processor is in charge of encoding a single view, this new model focuses on a flexible assignment of the encoding of frames among the pool of available processors. This new model is more suitable for realistic scenarios in which the number of views and processors is different. Moreover, we show that for this new encoder model, a more efficient use of the processors can be achieved by means of appropriate frame coding ordering algorithms. This is especially interesting in real-time encoding applications in which frame processing time constraints are imposed in order to maintain low encoding latency. Finally, we evaluate the latency performance of this new encoder model and show its advantages over the fixed encoder scheme.**

I. Introduction

3D Video (3DV) and Free Viewpoint Video (FVV) are new types of visual media that expand the user's experience beyond what is offered by 2D video [1]. 3DV offers a 3D depth impression of the observed scene, while FVV allows an interactive selection of the viewpoint and direction within a certain operating range. A common element of 3DV and FVV systems is the transmission of multiple views of the same scene to the user. Multiview Video Coding (MVC) [2] is an extension of the Advanced Video Coding (AVC) [3] standard that provides efficient coding of such multiview video.

In MVC a new prediction relationship between frames is introduced, adding interview prediction to the single-view approach of AVC. This way MVC allows a very flexible design of temporal and interview prediction dependencies. Different designs and implementations of this open architecture lead to considerably different coding performances. This makes it necessary to analyze the relative coding efficiency for different prediction structures. For example, Merkle *et al* [4] propose various efficient prediction structures.

In the literature, comparison between different multiview prediction structures has been mainly focused on rate-distortion (RD) performance. In [5] we argued that using solely RD performance ignores important differences between structures when considering the implementation of a multiview encoder, especially in applications such as multiview videoconferencing, where low latency is an important requirement and strict constraints on end-to-end delay are imposed. In [5] we provided a general framework for a systematic encoding latency analysis of arbitrary multiview prediction structures. This analysis framework was built upon a dependency graph extracted from the multiview prediction structure. This dependency graph is independent of the number of processors or the processor architecture, used to implement the multiview encoder. However, the software tool and the encoder model within that framework assume a multi-processor encoder architecture model in which all frames of a given view are encoded by the same processor, so that the number of processors is equal to the number of views. One drawback of that model is that it is not able to capture more realistic and flexible multiprocessor encoder architectures in which the number of views is not equal to the number of processors. We will refer to it as the *Fixed Multi-Processor Encoder* (Fixed MPE).

The *encoding latency* in a multiview encoder is the maximum delay (over all frames in the multiview sequence) between the capture of a frame and the instant when that frame is completely coded [5]. Encoding latency in the Fixed MPE model is dependent on the multiview prediction structure and the *frame processing time* (processing time devoted to the frame encoding operations). In this paper we argue that in order to establish a bounded encoding latency value (the encoding latency is bounded if the latency of any given frame is bounded) there is a maximum limit for the frame processing time. If this maximum time is exceeded the encoding latency is not bounded and increases over time, making the system not suitable for real-time encoding applications. In a Fixed MPE model, due to interview prediction, when this maximum is reached only some of the processors will be working at the maximum capacity, so that there will exist idle time periods for the remaining processors. Therefore, more flexible architectures can take advantage of this unused processor time in order to improve overall performance.

In this paper we extend the encoding latency analysis framework of [5] by providing models and tools that consider a *Flexible Multi-Processor Encoder* (Flexible MPE) architecture. In this architecture, the number of processors can be different from the number of views. Moreover, each processor is not statically assigned to one view, but the pool of available processors can encode frames from any of the input views. We will show that for this new encoder model the maximum frame processing time that guarantees a bounded latency, is higher than that of the Fixed MPE architecture. Therefore, higher RD performance coding algorithms could be implemented, especially in computation scalable encoders in which different encoding parameters and configurations can be selected to scale complexity, and rate-distortion-complexity optimization algorithms are used [6]. We also discuss that, for the Flexible MPE model, an algorithm to order the coding of the frames is needed as different coding orders may result in different latency values. Finally we evaluate the latency performance of this new scheme, including a comparison with the Fixed MPE model. Results will show that, for two encoders with equal number of processors with same characteristics, the Flexible MPE model makes better use of the processors achieving a higher value for the maximum frame processing time that is permitted. In the results we present an example in which, for the frames with highest coding complexity in the prediction structure, a 16-35% extra available processing time could be achieved.

This paper is organized as follows: in Section II we show the frame processing time limits for the Fixed MPE model and we present our new Flexible MPE model. In Section III we present experimental results on latency performance evaluation of the Flexible MPE model and we compare it against the latency performance of the Fixed MPE model. In Section IV we present the conclusions.

II. Multi-processor encoder model discussion

In [5] we presented the Fixed MPE model that considers a multi-processor architecture for a multiview encoder with N views and N processors. This model is characterized by the following restrictions:

- Each processor is assigned to a single view: processor i only encodes frames from view i and all frames from view i are encoded by processor i.
- The processors encode their assigned frames sequentially.
- If, at a given time, several frames in a view are ready to be coded (all their reference frames have been coded), these frames are encoded in display order (i.e., the frame to be displayed first will be coded first).

To make that scenario realistic, we modeled the frame processing time of frame x_j^i (frame j of view i), $\Delta t\text{proc}_j^i$, by assuming that it depends on the number of reference frames used to encode it:

$$\Delta t\text{proc}_j^i = \Delta t_{\text{basic}} + n(i,j)\Delta t_{\text{ref}}, \tag{1}$$

where Δt_{basic} is the time devoted for all the operations not related to motion estimation and motion compensation, $n(i,j)$ is the number of references for frame x_j^i and Δt_{ref} is the incremental processing delay required for each additional reference frame.

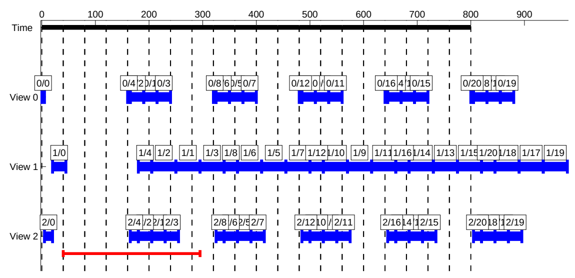

Fig. 1. Encoding chronogram for a Fixed MPE model implementation. Blue bars show the processing time for each of the frames. The red bar at the bottom shows the encoding latency of the whole sequence.

A. Bounded encoding latency value condition for the Fixed MPE model

For applications such as multiview video-conferencing, in which end-to-end constraints are imposed, it is necessary for the encoding latency to be bounded, i.e. encoding latency cannot grow monotonically with the number of coded GOPs, and therefore there exists a maximum encoding delay for all the frames in the multiview sequence. For the Fixed MPE model, we give here the condition that frame processing time must satisfy in order to ensure that this bound encoding latency value exists.

If we consider a single view encoder, this condition is satisfied when the sum of the frame processing time for each of the frames of the GOP is equal or lower than the total capture period of the whole GOP. Although the encoding latency may exceed the GOP capture period, due to backward predictions, if this condition holds latency does not increase for following GOPs. In our frame processing time model, that is:

$$\sum_{j=0}^{M_{\text{GOP}}-1} (\Delta t_{\text{basic}} + n(j)\Delta t_{\text{ref}}) \leq M_{\text{GOP}}\, T_{\text{capt}} \tag{2}$$

where M_{GOP} is the number of frames per GOP and T_{capt} is the frame capture period, i.e. the time elapsed between the capture of two consecutive frames.

For the Fixed MPE model, the encoding latency value is bounded when, the later condition holds for each of the views of the multiview set, i.e., for each view the sum of the frame processing time of every frame of the GOP, belonging to that view, is equal or lower than the capture time of the whole GOP. This is expressed formally by:

$$\mathbf{max}_i \left(\sum_{j=0}^{M_{\text{GOP}_v}-1} \Delta t\text{proc}_j^i \right) \leq M_{\text{GOP}_v}\, T_{\text{capt}}, \tag{3}$$

where M_{GOP_v} is the number of frames per GOP in a view, the index i indicates that frames belong to view i and $i = 1...N$.

Most multiview prediction structures have an unbalanced computational load, i.e., a few views may be more complex to encode than others due, for example, to the use of interview prediction. Frames on those views use extra interview reference frames, needing a higher processing time than frames in other views. In the Fixed MPE model, these views define a bottleneck for the bounded encoding latency condition. For a certain pair (Δt_{basic}, Δt_{ref}) that defines the frame processing time limit, the expression in (3) is satisfied with equality for only some of the views. Thus, for the other views some of the processors are used below their maximum capacity, i.e., during certain time periods of the encoding process, some of these processors are idle.

Figure 1 shows the encoding chronogram [5] for a JMVM [7] prediction structure of three views and a GOP size of 4 frames. While the processor assigned to View 1 is fully used, the processors assigned to View 0 and View 2 are being used below their maximum capabilities, i.e. idle time periods exist for processors assigned to View 0 and View 2.

This gives the intuition that using a different encoder architecture, in which frames can be coded in any of the processors, a better distribution of the computational load can be achieved, and therefore processing time for each frame could be increased (implementing coding algorithms with higher RD efficiency) while maintaining a bounded encoding latency value. Moreover, an encoder model with these characteristics includes a wider range of encoder architectures as it permits a different number of processors and views.

B. Flexible multi-processor encoder model

In order to extend the multiview encoder model, we present here the Flexible MPE model that considers a multiview encoder with N views and K processors, with the following characteristics:

- Each processor is **not** assigned to a single view: any frame from any of the N views can be encoded in any of the K processors.
- The processors encode their assigned frames sequentially.
- If, at a given time, several frames are ready to be coded and K_f out of K processors are free:
  - First, these frames are ordered by a frame priority measure.
  - Then, the first K_f frames in the frame order list are encoded by assigning each one of them to one of the K_f free processors.

In this new architecture model we maintain the frame processing time model, considering in addition, that Δt_{basic} includes a processing time dedicated to the frame data exchange from the capture module to its assigned processor. Depending on the encoder implementation, this data exchange time may differ from the data exchange time in the Fixed MPE model (in a software-based platform both times should be comparable).

C. Coding order based on frame priority

For this type of encoder model, the coding order decisions are not straightforward. Although some restrictions in the coding order are imposed by the prediction structure (a frame cannot be encoded before its reference frames), there are still some choices to be made about the coding order, and these can lead to different encoding latency values.

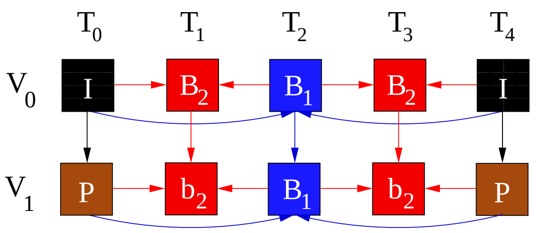

Fig. 2. GOP prediction structure for two views.

Consider, as an example, the prediction structure in Figure 2, and assume that at the time instant immediately after frame V_0/T_2 has been coded only one of the processors is free. Then, frames V_0/T_1 and V_0/T_3 are ready to be coded. Frame V_0/T_1 should be coded first in order to minimize the maximum latency for both frames and therefore for the whole sequence. But capture order should not be the only criterion due to interview prediction. Consider now the time instant immediately after V_0/T_4 has been coded. If the capture time is the only criterion, frames V_0/T_1, V_0/T_2 and V_0/T_3 would be coded before frame V_1/T_4, increasing the encoding latency for frames of V_1, and the maximum of the latencies for the whole GOP. In our model we also take into account the number of frames that have a reference dependency on a certain frame to decide its coding order.

At time t, and given a set of frames that are ready to be coded, the frames are ordered in a coding order list, following these ordering rules:

- Frames from earlier GOPs are coded first.
- Frames within the same GOP are ordered by a frame priority value λ. Frames with higher priority are coded first.

The priority value λ_j^i for frame x_j^i is calculated using the following expression:

$$\lambda_j^i = (t - t_{\text{capt}_j^i}) + w\lambda_{\{\mathcal{K}(x_j^i)\}}, \tag{4}$$

where $(t - t_{\text{capt}_j^i})$ is the difference between the actual time instant and the capture time of the frame while $\lambda_{\{\mathcal{K}(x_j^i)\}}$ directly depends on the number of frames in the GOP that require frame x_j^i to be coded before they can be coded themselves. The value $\lambda_{\{\mathcal{K}(x_j^i)\}}$ is given by:

$$\lambda_{\{\mathcal{K}(x_j^i)\}} = \sum_{k \in \{\mathcal{K}(x_j^i)\}} \alpha_k (t - t_{\text{capt}_k}), \tag{5}$$

where $\{\mathcal{K}(x_j^i)\}$ is the set of frames that use frame x_j^i as reference directly and indirectly, i.e. all the frames that are in a higher dependency level than x_j^i in the dependency graph [5] and such that there exists a decoding path to this frame from x_j^i. α_k is 1 if frame x_k has been captured already and 0 otherwise.

In order to trade-off the two terms in λ_j^i we use the weight factor w. For the analyzed prediction structures, we have empirically obtained that the value of w that minimize the

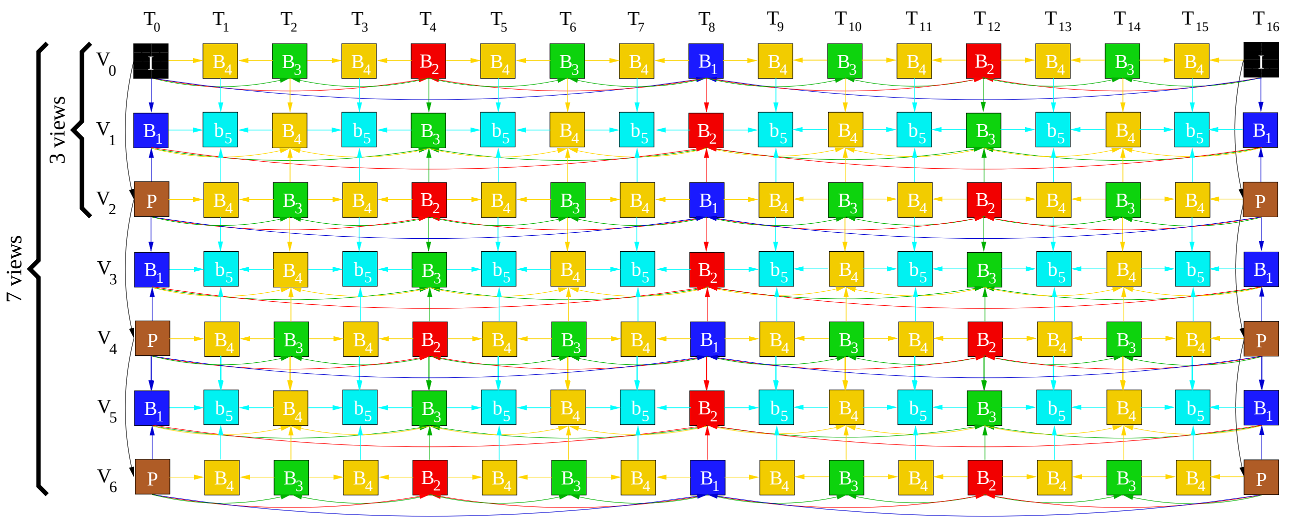
T$_0$ T$_1$ T$_2$ T$_3$ T$_4$ T$_5$ T$_6$ T$_7$ T$_8$ T$_9$ T$_{10}$ T$_{11}$ T$_{12}$ T$_{13}$ T$_{14}$ T$_{15}$ T$_{16}$
3 views
7 views
V$_0$ I B$_4$ B$_3$ B$_4$ B$_2$ B$_4$ B$_3$ B$_4$ B$_1$ B$_4$ B$_3$ B$_4$ B$_2$ B$_4$ B$_3$ B$_4$ I
V$_1$ B$_1$ b$_5$ B$_4$ b$_5$ B$_3$ b$_5$ B$_4$ b$_5$ B$_2$ b$_5$ B$_4$ b$_5$ B$_3$ b$_5$ B$_4$ b$_5$ B$_1$
V$_2$ P B$_4$ B$_3$ B$_4$ B$_2$ B$_4$ B$_3$ B$_4$ B$_1$ B$_4$ B$_3$ B$_4$ B$_2$ B$_4$ B$_3$ B$_4$ P
V$_3$ B$_1$ b$_5$ B$_4$ b$_5$ B$_3$ b$_5$ B$_4$ b$_5$ B$_2$ b$_5$ B$_4$ b$_5$ B$_3$ b$_5$ B$_4$ b$_5$ B$_1$
V$_4$ P B$_4$ B$_3$ B$_4$ B$_2$ B$_4$ B$_3$ B$_4$ B$_1$ B$_4$ B$_3$ B$_4$ B$_2$ B$_4$ B$_3$ B$_4$ P
V$_5$ B$_1$ b$_5$ B$_4$ b$_5$ B$_3$ b$_5$ B$_4$ b$_5$ B$_2$ b$_5$ B$_4$ b$_5$ B$_3$ b$_5$ B$_4$ b$_5$ B$_1$
V$_6$ P B$_4$ B$_3$ B$_4$ B$_2$ B$_4$ B$_3$ B$_4$ B$_1$ B$_4$ B$_3$ B$_4$ B$_2$ B$_4$ B$_3$ B$_4$ P

Fig. 3. JMVM multiview prediction structures with three and seven views.

encoding latency is:

$$w = \frac{1}{d_{max,b}}, \tag{6}$$

where $d_{max,b}$ is the maximum distance (in number of frames) between a frame and the furthest frame with which it has a direct or indirect backward dependency relationship. As an example, distance between V_0/T_1 and V_0/T_4 in Figure 2.

III. Results

To evaluate the latency performance of the Flexible MPE model, we have carried out two different tests: first, we have evaluated the latency performance of the proposed encoder model varying the number of processors and the frame processing time. As a second test, we have compared the latency performance of the Flexible MPE model and the Fixed MPE one. We have compared two multiview encoders with the same number of processors (equal to the number of views) one following the Flexible MPE model and the other following the Fixed MPE model.

For the experiments, we have used two different multiview prediction structures, a JMVM GOP structure of three views and a size of 16 frames and a JMVM GOP structure of 7 views and a size 16 frames. Both prediction structures are shown in Figure 3. In both tests we have focused on evaluating the frame processing time limits that ensure a bounded value of the encoding latency and the evolution of the latency value with the value of the frame processing time.

A. Evaluation of Flexible MPE model encoding latency performance

A comparative encoding latency evaluation for a set of multiview encoders within the Flexible MPE model and different number of processors has been performed. We have evaluated the latency performance of six encoders using one to six processors when encoding a multiview sequence with the same JMVM prediction structure of three views and GOP

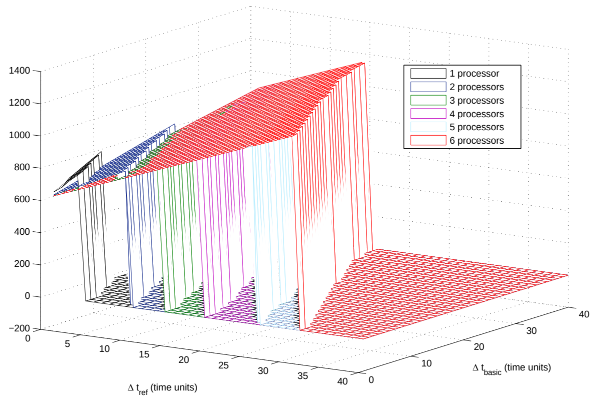

Fig. 4. Latency performance for a Flexible MPE model with different number of processors.

size of 16 frames. Figure 4 shows the encoding latency values for different number of processors and different values of the frame processing time parameters Δt_{basic} and Δt_{ref}. When, for a given pair of frame processing time parameter values, the encoding latency does not have a bounded value (encoding latency depends on the number of coded GOPs), this is represented with a null latency value in the Figure. Figure 5 shows the processing time limits for all the encoder implementations. Valid values for frame processing time parameters for each number of processors are those in the area below each graph. Finally, to show the results in Figure 4 in a clearer way, Figure 6 shows the latency performance for all the encoders when equal values for both processing time parameters are used. In the same way, for this Figure, if for a given frame processing time value the encoding latency is unbounded, this is represented with a null latency value.

The results in Figure 5 show that increasing the number of processors leads to a increase in the processing time that can

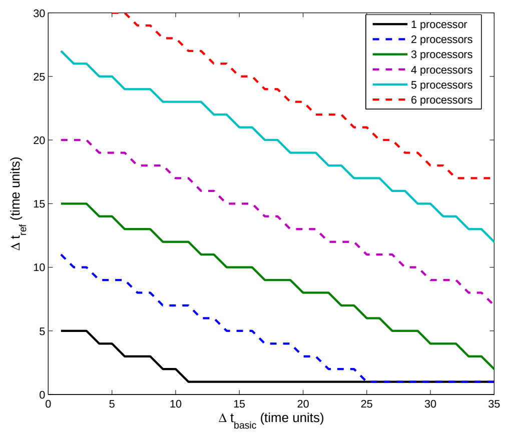

Fig. 5. Processing time limits for a Flexible MPE model with different number of processors.

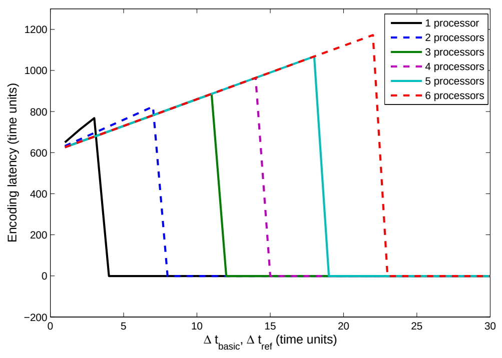

Fig. 6. Latency performance for a Flexible MPE model with different number of processors and equal value for both frame processing time parameters.

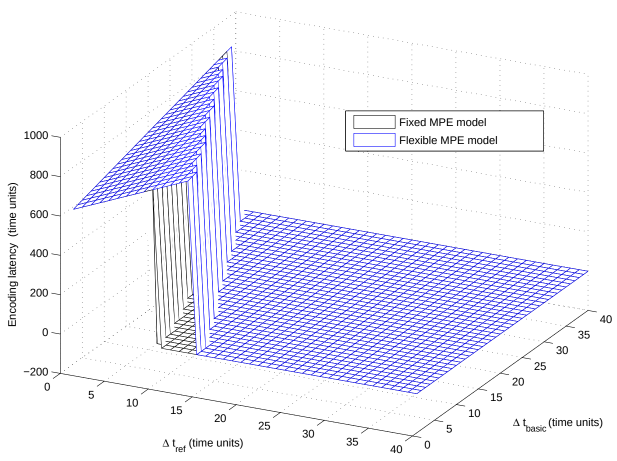

Fig. 7. Latency performance for both Fixed and Flexible MPE models. Data for three views and three processors.

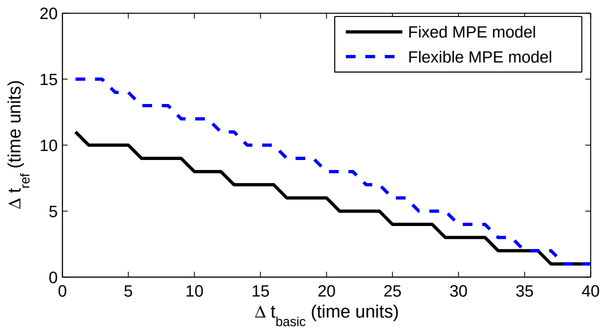

Fig. 8. Processing time limits for both Fixed and Flexible MPE models. Data for three views and three processors.

be devoted to the encoding process of each frame. Results in Figures 4 and 6 show that, for a given pair of values of the frame processing time parameters, a higher number of processors results in a lower latency value until the number of processors is equal or higher than the number of views. Using a number of processors above this limit increases the capacity of the system to dedicate more time to frame encoding but does not decrease the encoding latency value for this type of prediction structures.

B. *Fixed and Flexible MPE model comparison*

In this comparison we show that, for two multiview encoders with the same number of processors, and the same prediction structure, one within the Fixed MPE model and the other within the Flexible MPE model, for the second one the frame processing time limits are higher. We have evaluated the encoding latency performance of both models for a JMVM prediction structure of three views and a GOP size of 16 frames, as the one depicted in Figure 3. The results are shown in Figure 7.

Figure 8 shows the processing time limits for both encoder models and Figure 9 shows the latency performance using equal values for both frame processing time parameters Δt_{basic} and Δt_{ref}. The results show that using a Flexible MPE model higher frame processing times can be devoted to the encoding of the frames while maintaining a bounded latency value. When the frame processing time parameters are under the Fixed MPE model limits the latency performance of both encoder models is equal.

We have also compared the latency performance of the Fixed and Flexible MPE models using a higher number of views and processors. Figures 10 and 11 show the results for a multiview prediction structure with 7 views such as the one shown in Figure 3. The results show that there is only an insignificant latency increase (less than 1%) for the Flexible MPE model with respect to the Fixed MPE model for some of the parameter frame processing time parameter values. On the other side, results in Figure 11 show that for a $\Delta t_{\text{ref}}/\Delta t_{\text{basic}}$ ratio from 0.5 to 2 (it depends on the encoder implementation), we can note as an example, that for the B frames using four references (Figure 3) a 16% to 35% of additional available coding time could be achieved.

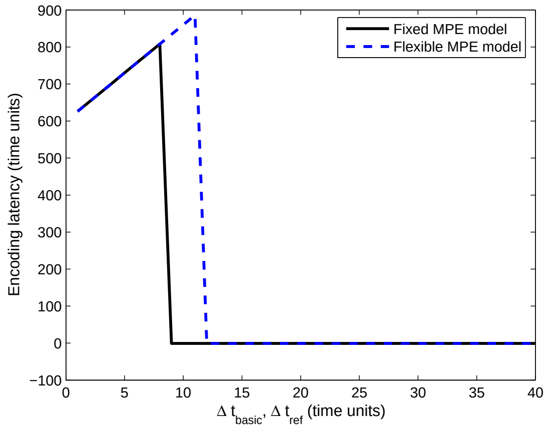

Fig. 9. Latency performance for both Fixed and Flexible MPE models with equal value for both processing time parameters. Data for three views and three processors.

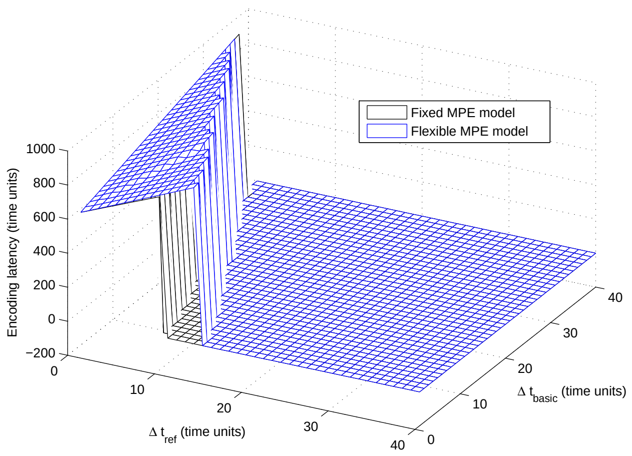

Fig. 10. Latency performance for both Fixed and Flexible MPE models. Data for 7 views and 7 processors.

IV. Conclusions

With the focus of real-time encoding applications, we have discussed the frame processing limits for a Fixed MPE model that ensure a bounded encoding latency value and some considerations about the processor idle time periods in this encoder scheme. Then, we have presented the characteristics of our new Flexible MPE model, which through a more balanced use of processors, reduces these idle times. For this new model, we have argued that, as different coding orders may result in different encoding latency performances, a coding order algorithm is needed. Therefore, we have proposed a coding order algorithm based on a frame priority value that takes into account the precedence of the frames in time and their prediction relationships.

Our experimental results show the evolution of the latency performance of the Flexible MPE model with the number of processors and the frame processing time. Through a latency performance comparison with the Fixed MPE model, for commonly used multiview prediction structures, we show that our new encoder model makes a better use of the pool of available processors, achieving higher values for the maximum frame processing time that is permitted.

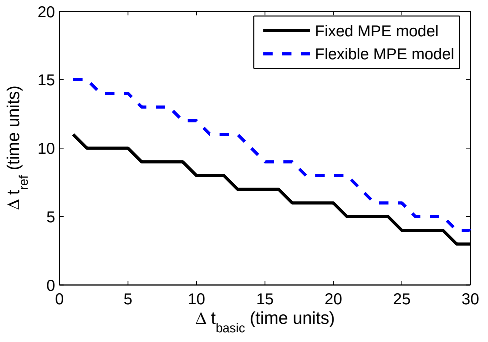

Fig. 11. Processing time limits for both Fixed and Flexible MPE models. Data for 7 views and 7 processors.

Acknowledgment

This work has been partially supported by SAPEC and the Spanish Administration agency CDTI under project CENIT-VISION 2007-1007, by the Ministerio de Ciencia e Innovación of the Spanish Government under project TEC2007-67764 (SmartVision), and by the Comunidad de Madrid under project S-0505/TIC-0223 (Pro-Multidis).

References

[1] A. Smolic, K. Mueller, P. Merkle, C. Fehn, P. Kauff, P. Eisert, and T. Wiegand, “3D Video and Free Viewpoint Video - Technologies, Applications and MPEG Standards,” *Proc. IEEE International Conference on Multimedia and Expo 2006* , pp. 2161–2164 (July 2006).
[2] A. Vetro, P. Pandit, H. Kimata, A. Smolic, and Y. Wang, “Joint Draft 8.0 on Multiview Video Coding,” *Doc. JVT-AB204 Hannover, Germany* (July 2008).
[3] ITU-T Rec. & ISO/IEC, “14496-10 AVC Advanced Video Coding for Generic Audiovisual Services,” (2005).
[4] P. Merkle, A. Smolic, K. Mueller, and T. Wiegand, “Efficient Prediction Structures for Multiview Video Coding,” *IEEE Transactions on Circuits and Systems for Video Technology* **17**, pp. 1461–1473 (Nov. 2007).
[5] P. Carballeira, J. Cabrera, A. Ortega, F. Jaureguizar and N. Garcia, “Comparative Latency Analysis for Arbitrary Multiview Video Coding Prediction Structures,” *IS&T/SPIE Int. Conf. on Visual Communications and Image Processing, VCIP 2009*, San Jose (CA), USA, SPIE vol. 7257, pp. 72570L-1-12, 18-22 Jan. 2009.
[6] D.S. Turaga, M. van der Schaar, and B. Pesquet-Popescu, “Complexity scalable motion compensated wavelet video encoding”, *IEEE Transactions on Circuits and Systems for Video Technology*, vol.15, no.8, pp. 982-993, Aug. 2005.
[7] A. Vetro, P. Pandit, H. Kimata, A. Smolic, and Y. Wang, “Joint Multiview Video Model (JMVM) 8.0,” *Doc. JVT-AA207 Geneva, Switzerland* (April 2008).