A WORKLOAD BALANCED PARALLEL VIEW SYNTHESIS FOR FTV

Zhanqi Liu, Xin Jin, Chenyang Li and Qionghai Dai

Shenzhen Key Lab of Broadband Network and Multimedia,
Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China

ABSTRACT

In this paper, a parallel system together with an adaptive workload balancing algorithm is proposed for view synthesis on multi-core platforms. Based on system level data parallelism, an adaptive workload balancing method is proposed for depth image based rendering by evaluating the number of non-hole pixels after warping. Experimental results demonstrated that with the proposed workload balancing algorithm, the workload difference among the cores is reduced by 90.65% on average for 2-core systems and by 79.57% on average for 4-core systems, respectively. Compared with the parallel system without the proposed balancing algorithm, synthesis speed is further improved by 7.5% for 2-core systems and 8.9% for 4-core systems at maximum, respectively, without degradation in the subjective and objective quality.

Index Terms—Parallel view synthesis, depth image based rendering, workload balancing, FTV

1. INTRODUCTION

As the future of 3DTV, free viewpoint TV (FTV) has attracted great attention both from industry and academy. It allows viewers to watch a 3D scene by freely changing the viewpoints [1]. The Moving Picture Experts Group (MPEG) started FTV standardization project in April 2007 and has adopted a system architecture of performing depth estimation at the sender and view synthesis at the receiver with the data format Multiview Video plus Depth (MVD) [2].

The MVD based architecture is feasible for FTV implementation using limited input data, while it also introduces big computational burden at the receiver, where the virtual views need to be generated by depth image-based rendering (DIBR) [3]. In order to obtain virtual views with good quality, DIBR introduces Warping, Blending, Hole Filling and Boundary Noise Removing [3] to map the reference views to the virtual viewpoint and to correct the pixel values accordingly. However, the process presents high computational complexity, which affects the application of FTV especially on portable devices, such as smart phones, tablet PCs etc.

Recently, some works have been proposed for fast view synthesis based on GPU [4][5]. However, for multi-core systems, which are widely applied in portable devices, to the best of our knowledge, no parallel view synthesis algorithm has been proposed yet. Furthermore, for the parallel applications, workload balancing among the cores is essentially important. The efficiency of parallelism is mainly constrained by the execution time of the core with the highest workload [6]. Developing a parallel view synthesis system with balanced workload is fundamentally required by the portable devices for FTV immigration.

Consequently, a parallel view synthesis system with an adaptive workload balancing algorithm is proposed in this paper. The system supports system level data parallelism by partitioning the reference views according to the requirements from the hardware and the users. The proposed workload balancing algorithm extracts the number of non-hole pixels from the warped image to predict the partition accurately. Experiments results demonstrated that the proposed workload balancing algorithm introduces an average reduction of 90.65%/79.57% in workload balance difference, which leads to the 1.89/3.47 times speedup ratio achieved by the whole system on 2-core/4-core systems. Negligible degradation in objective quality is observed in the synthesized views, which is not detectable subjectively.

The rest of the paper is organized as follows. In Section 2, the proposed parallel view synthesis system is introduced. In Section 3, the workload balancing algorithm is described in detail. The experimental results are provided in Section 4 with conclusions in Section 5.

2. PARALLEL VIEW SYNTHESIS SYSTEM

The system architecture of the proposed parallel view synthesis with workload balancing is depicted in Fig. 1. As shown in the figure, the system mainly consists of *Data Partitioning*, *View Synthesis*, *Adaptive Workload Balancing* and *Data Integrating*.

In *Data Partitioning*, system level data parallelism is applied to partition each frame of the reference views according to the number of cores or the requirements from the users. Each frame can be partitioned horizontally, vertically, or partitioned into tiles like that defined in 3D-HEVC [2]. Texture and depth from the left and right reference views take the same partition scheme. The size of the partitions is determined by *Adaptive Workload Balancing*. Each partition group, including texture and depth partitions from the left and right reference views, is assigned to a core for view synthesis.

In *View Synthesis*, both 1-D and 2-D view synthesis [8] techniques based on DIBR can be applied, which is determined by the arrangement of cameras and the location of virtual views. A partition of the virtual view will be generated after *View Synthesis*, which will be sent to *Data Integrating* for merging.

Data Integrating gathers the synthesized partition from each core. It waits until all the cores finish view synthesis and combines the synthesized partitions into a complete virtual view for output. To be noted that if the computational workload among cores is quite different, the efficiency of parallel synthesis will be greatly reduced because of waiting. Consequently, an *Adaptive Workload Balancing* is proposed to balance the workload among the cores.

Adaptive Workload Balancing balances the workload based on the non-hole pixels distribution in the warped image. It predicts the future workload by retrieving the number of non-hole pixels in each line from the warped image and determines the partition size of the next frame based on the predicted workload. The size of the partitions adapts to video contents and the maximum disparity in the frame. The proposed workload balancing algorithm will be described in detail in Section 3.

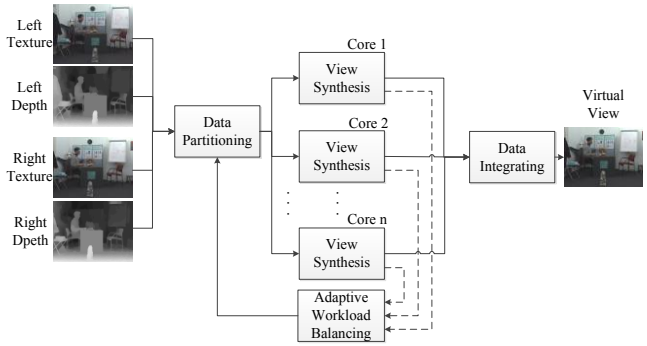

Fig. 1. The proposed parallel view synthesis system.

3. ADAPTIVE WORKLOAD BALANCING

Adaptive Workload Balancing determines the size of the partitions by temporal prediction. Considering the most general 3D applications uses horizontally arranged cameras [1], and partitioning the frame horizontally introduces an average of 0.4dB increment in PSNR for the synthesized views relative to partitioning vertically and partitioning into tiles because of exploiting more neighboring pixels, the following derivations are proposed for horizontal partitioning based on 1-D view synthesis [8]. Nevertheless, all the derivations can be extended to other partition types easily.

3.1. Definition of workload

Generally, the normalized workload of a task can be measured by the processor utilization, which is proportional to the execution time of a task [7]. Boundary Noise Removing is discarded for that it doesn't provides a stable performance in objective quality even resulting in a significantly worse in some content. So, experiments have been carried out to retrieve the execution time for the other three major processes in 1-D view synthesis: Forward Warping, Merging and Hole Filling. It is observed that Forward Warping and Merging occupy around 94% of the total execution time of view synthesis. Thus, the workload of view synthesis for a partition i, W_i, can be approximated by

$$W_i \approx W_{F,i} + W_{M,i} \ , \tag{1}$$

where $W_{\mathrm{F,i}}$ and $W_{\mathrm{M,i}}$ are workload of Forward Warping and Merging, respectively.

Forward Warping warps the reference partition to the virtual viewpoint pixel by pixel according to the 3D warping formula [3]. So, its workload is proportional to the partition size, which is given by

$$W_{F,i} = 2\alpha_1 w_i h_i \ , \tag{2}$$

where α_1 denotes the warping workload per pixel; w_{i} and h_{i} denote the width and height of partition i, respectively. Multiplying by 2 represents warping from the left and right views simultaneously.

Merging blends the two partitions warped from the left and right reference partitions and generates the partition at the targeting virtual viewpoint pixel by pixel. Since it processes the pixel positions in the non-hole regions either warped from the left view or the right view, the workload of Merging, $W_{M,i}$, is proportional to the total number of pixels in the non-hole regions. So, it is given by

$$W_{M,i} \approx \alpha_2 \sum_{j=1}^{h_i} (N_{l,i}[j] + N_{r,i}[j]) \ , \tag{3}$$

where α_2 denotes the blending workload of a pixel in non-hole regions; $N_{l,i}[j]$ and $N_{r,i}[\mathrm{j}]$ denote the total number of non-hole pixels in line j of the warped partition i from the left and right views, respectively.

Since α_1 and α_2 are algorithm and platform dependent, they are retrieved and updated dynamically with view synthesis for a stable result on a specific platform. Therefore, α_1 is given by

$$\alpha_1 \approx \sum_N \left(\sum_{i=1}^{n} W_{F,i} \right) \Big/ (2NWH) \ , \tag{4}$$

where N is the total number of processed frames; n is the total number of cores; W and H represents the image height and width, respectively. Similarly, α_2 is given by

$$\alpha_2 \approx \frac{\sum_N \sum_{i=1}^{n} W_{M,i}}{\sum_N \sum_{i=1}^{n} \sum_{j=1}^{h_i} (N_{l,i}[j] + N_{r,i}[j])} \ . \tag{5}$$

3.2. Workload balancing

To balance the workload among cores, workload overloading of each core is measured by comparing with the targeting average workload:

$$\Delta W_i = W_i - \frac{1}{n} \sum_{i=1}^{n} W_i \ . \tag{6}$$

ΔW_i larger than zero represents the workload assigned to core i is higher than average, which needs to be offloaded. Inversely, lower than zero represents more workload can be assigned.

Then, the workload among the cores can be adjusted in a recursive way. Taking horizontal partitioning as an instance, the height of partition 1, h_1, can be determined to compensate the workload difference ΔW_1. Then, the height of partition 2, h_2, can be adjusted to compensate $\Delta W_1+\Delta W_2$ due to the variation in h_1. The process can be carried on until all the partitions are adjusted. Thus, for h_k (k=1,2,…,n-1), if the corresponding workload difference is positive/negative, the partition height is reduced/increased by searching upward/downward line by line until

$$2\alpha_1 w_m \Delta h + \alpha_2 \sum_{j=h_k}^{h_k+\Delta h} (N_{l,m}[j]+N_{r,m}[j]) \geq \left| \sum_{i=1}^{k} \Delta W_i \right| ,$$

$$m = \begin{cases} k, \ \sum_{i=1}^{k} \Delta W_i > 0 \\ k+1, \ \sum_{i=1}^{k} \Delta W_i < 0 \end{cases} . \quad (7)$$

The obtained Δh will be used to update the partition size h_k for the next frame by

$$h_k^{new} = \begin{cases} h_k - \Delta h, \ \sum_{i=1}^{k} \Delta W_i > 0 \\ h_k + \Delta h, \ \sum_{i=1}^{k} \Delta W_i < 0 \end{cases} . \quad (8)$$

If the color components associated with the texture follows 4:2:0 sampling format, one will be added to h_k^{new} if h_k^{new} is an odd value.

3.3. Overhead analysis

For the storage complexity overhead introduced by the proposed workload balancing algorithm: recording the total number of non-hole pixels in each line of the left and right warped images introduces maximum $2\times H\times \log_2(W)/8$ bytes. Taking the video with the resolution of 1920×1080 as an instance, 3.2K bytes are needed, which is negligible to implementation.

For the computational complexity overhead introduced by the proposed workload balancing algorithm, only nine additions and one comparison are needed for each line's adjustment. Experiment results reveal that the proposed algorithm only takes up an average of 0.012% of the total execution time, which is feasible for its implementation on portable devices.

4. EXPERIMENTAL RESULTS

The proposed parallel system is integrated into reference software, VSRS 3.5 [8], to test its performance. The experiments are conducted on a PC with an Intel®Core™ i5-3470 3.2GHz quad-core processor and 12 GB RAM under 64-bit Windows 7. 1-D mode view synthesis with half-pixel precision is selected for VSRS, while other options are set to the default values [8]. Four sequences with representative features in content and disparity are selected for testing, as listed in Table 1. "Syn. View" represents the virtual view synthesized by the input views. For simplicity, the name of sequences will be abbreviated by the first three characters in the following.

Table 1. Test sequences and viewpoints.

Seq.	Res.	No. of Frames	Input Views	Syn. View
Bookarrival [9]	1024x768	100	10, 8	9
Champagne_tower [10]	1280x960	100	39, 41	40
Newspaper [9]	1024x768	100	4, 6	5
PoznanStreet [11]	1920x1088	100	3, 5	4

4.1. The proposed system without workload balancing

The performance of proposed parallel system without workload balancing (denoted by *Propw*/*oWB*), is compared with that of single-thread VSRS (denoted by *VSRS*) in Table 2, where 2-thread (n=2) and 4-thread (n=4) parallel synthesis cases are evaluated. As shown in the table, *Propw*/*oWB* accelerates the view synthesis process by an average of 1.8/3.25 times under 2-thread/4-thread case. Meanwhile, only 0.007/0.003 dB degradation in the synthesis quality is introduced, which is negligible to real applications and presents no influence on the visual quality. The slight quality degradation is mainly caused by the reduction in the pixel correlations around the partition boundaries.

Table 2. Synthesis speed and quality compared with *VSRS*.

Seq.	Speedup Ratio		ΔPSNR(dB)	
	$n=2$	$n=4$	$n=2$	$n=4$
Boo.	1.82	3.31	-0.026	0.002
Cha.	1.81	3.16	-0.001	0.001
New.	1.74	3.24	0.001	-0.001
Poz.	1.83	3.30	-0.002	-0.013
Ave.	1.80	3.25	-0.007	-0.003

4.2. The proposed system with workload balancing

The performance of proposed parallel system with workload balancing (denoted by *Propw*/*WB*), is compared with that of *Propw*/*oWB* in Table 3. The performance of workload balancing is measured by the average reduction ratio in the normalized maximum workload difference among the cores, which is given by

$$WDRR = 1 - \frac{1}{N}\sum_{N} \frac{\left(\max_{i,j\in[1,n]} \left| W_{i,\mathrm{WB}} - W_{j,\mathrm{WB}} \right| \right) \Big/ \left(\max_{i\in[1,n]} W_{i,\mathrm{WB}} \right)}{\left(\max_{i,j\in[1,n]} \left| W_{i,\mathrm{nWB}} - W_{j,\mathrm{nWB}} \right| \right) \Big/ \left(\max_{i\in[1,n]} W_{i,\mathrm{nWB}} \right)} , \quad (9)$$

where $W_{i,\mathrm{nWB}}$ and $W_{i,\mathrm{WB}}$ are the workload of the ith partition processed in *Propw*/*oWB* and *Propw*/*WB* (without and with workload balancing), respectively. So, the larger *WDRR* is, the larger the improvement in workload balancing among the cores is achieved. The workload is measured by the execution time. As shown in the table, *Propw*/*WB* improves workload balancing among the cores by an average of 90.65%/79.57% together with an additional 1.05/1.07 times acceleration in the synthesis speed for 2-thread/4-thread

cases, respectively. Also, it keeps the same or even better synthesis quality than that of *Propw/oWB*. Ultimately, *Propw/WB* outperforms *VSRS* by an average of 1.89/3.47 times in synthesis speed with only a quality loss of 0.003/0.005 dB for 2-thread/4-thread cases, respectively.

Fig. 2 and 3 compares of the size of partitions and the workload assigned to each core for sequence "Bookarrival" and "Champagne_tower", respectively. "Bookarrival" and "Champagne_tower" are different in resolution and background feature. As that demonstrated in the figures, with the workload balancing algorithm, the partition size will be adjusted adaptively with video content and the workload among the cores is balanced during the process.

Table 3. Comparison between Propw/WB and Propw/oWB.

Seq.	*WDRR* (%)		Speedup Ratio		ΔPSNR (dB)	
	n=2	n=4	n=2	n=4	n=2	n=4
Boo.	89.26	81.72	1.03	1.07	0.014	0.000
Cha.	91.81	81.12	1.03	1.05	0.005	0.003
New.	91.05	78.55	1.07	1.06	-0.008	-0.010
Poz.	90.48	76.67	1.07	1.09	0.003	-0.002
Ave.	90.65	79.57	1.05	1.07	0.004	-0.002

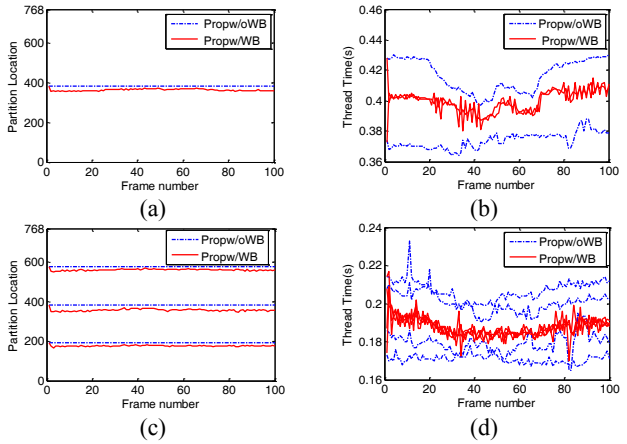

Fig. 2. Bookarrival sequence: (a) and (b) partition size and workload for 2-thread case; (c) and (d) partition size and workload for 4-thread case.

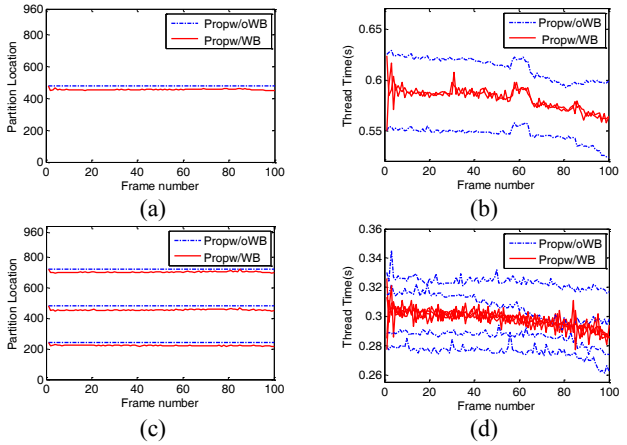

Fig. 3. Champagne_tower sequence: (a) and (b) partition size and workload for 2-thread case; (c) and (d) partition size and workload for 4-thread case.

Fig. 4 compares the subjective quality between *Propw/WB* and *VSRS* for Bookarrival and Champagne_tower. As shown in the figure, no visual quality difference can be detected in both 2-thread and 4-thread cases. No quality degradation can be observed visually.

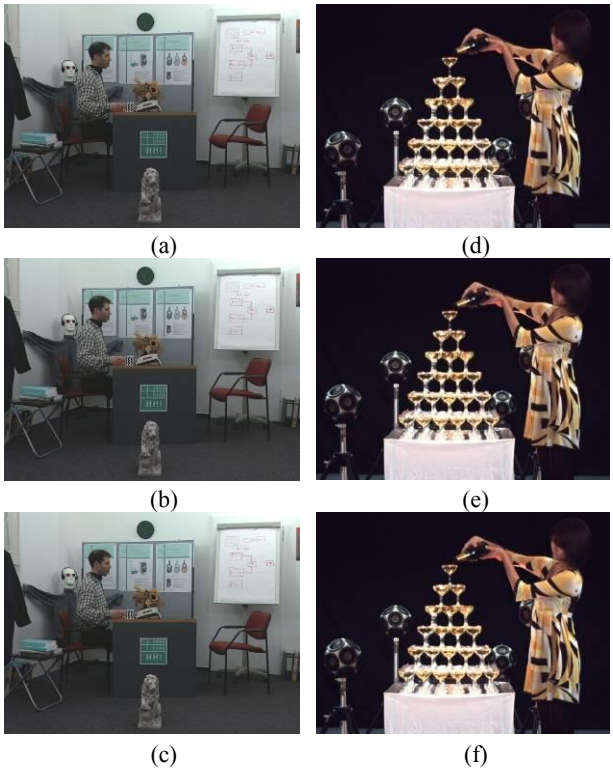

Fig. 4. Frame 1 of synthesized view 9 of Bookarrival generated by: (a) *VSRS*; (b) and (c) Propw/WB under 2-thread and 4-thread, respectively; Frame 1 of synthesized view 40 of Champagne_tower generated by: (d) *VSRS*; (e) and (f) Propw/WB under 2-thread and 4-thread, respectively.

5. CONCLUSIONS

Targeting multi-core platforms, a workload balanced parallel view synthesis system for FTV is proposed in this paper, which balances the workload using the distribution of non-hole pixels after warping. The simulation results show that the proposed parallel view synthesis system outperforms *VSRS* by 1.96/3.59 times at maximum in synthesis speed with the reality that the workload difference among the cores is reduced by 91.81%/81.72% after workload balancing for 2-core/4-core systems, respectively. Also, both the subjective and objective quality of the synthesized view are not degraded, which is very beneficial to fast view synthesis on multi-core platforms.

6. ACKNOWLEDGMENT

This work was supported in part by the NSFC-Guangdong Joint Foundation Key Project (U1201255) and project of NSFC 61371138, China.

7. REFERENCES

[1] M. Tanimoto, M.P. Tehrani, T. Fujii, and T. Yendo, "Free-Viewpoint TV," *Signal Processing Magazine*, *IEEE*, vol. 28, no. 3, pp. 67-76, 2011.

[2] G. Tech, K. Wegner, Y. Chen, and S. Yea, "3D-HEVC test model 5," in *JCT-3V Doc. JTC3V-E1005, 5th Meeting*, 2013.

[3] C. Fehn, "A 3D-TV approach using depth-image-based rendering (DIBR)," *Proc. of VIIP*, vol. 3, pp. 84-88, 2003.

[4] J.I. Jung, and Y.S. Ho, "Parallel view synthesis programming for free viewpoint television," *Signal Processing, Communication and Computing (ICSPCC)*, *2012 IEEE International Conference on*, pp. 88-91, 2012.

[5] H.C. Shin, Y.J. Kim, H. Park, and J.I. Park, "Fast view synthesis using GPU for 3D display," *Consumer Electronics, IEEE Transactions on* 54*,* no. 4, pp.2068-2076, 2008.

[6] K.H. Sihn, H. Baik, J.T. Kim, S. Bae, and H.J. Song, "Novel approaches to parallel H.264 decoder on symmetric multicore systems," in *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on,* pp.2017-2020, 2009.

[7] Sinha, Amit, and A.P. Chandrakasan, "Dynamic voltage scheduling using adaptive filtering of workload traces," *VLSI Design, 2001. Fourteenth International Conference on, IEEE*, pp. 221–226, 2001.

[8] View Synthesis Reference Software (VSRS 3.5) in *Tech. Rep. ISO/IEC JTC1/SC29/WG11*, 2010.

[9] http://sp.cs.tut.fi/mobile3dtv/stereo-video/.

[10] MPEG-FTV Test Sequence, available at: http://www.fujii.nuee.nagoya-u.ac.jp/~fukushima/mpegftv/.

[11] D. Rusanovskyy, K. Müller, and A. Vetro, "Common test conditions of 3DV core experiments," in *JCT-3V Doc. JCT3V-D1100, 4th meeting,* 2013.