GPU-Oriented Light Field Compression for Real-Time Streaming

Toru Ando* Yuichi Taguchi Takeshi Naemura
Graduate School of Information Science and Technology, The University of Tokyo

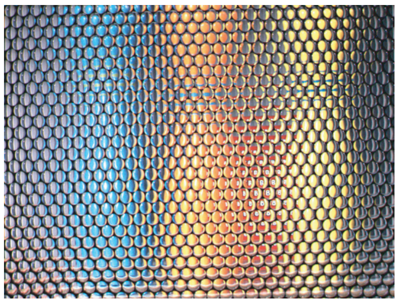

Figure 1: *An input integral photography*

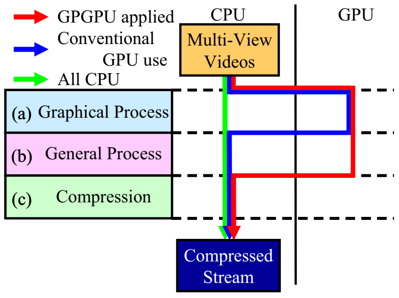

Figure 2: *Flow of encoding process*

Table 1: *Frame-rates for encoding/decoding*

Method	Encode	Decode
All CPU	0.94 fps	1.01 fps
Conv. GPU use	8.05 fps	23.47 fps
GPGPU applied	16.86 fps	24.60 fps

Table 2: *PC Configuration*

CPU	Pentium4 3.80GHz, 2.0GB RAM
GPU	GeForce6800, 256MB RAM

1 Introduction

Emerging applications such as free-viewpoint video and 3D-TV can enhance our viewing experience by rendering arbitrary viewpoint images using transmitted light field data. Compression is one of the key technologies for such systems due to the huge amount of data, typically captured with hundreds of cameras or thousands of lenslets.

We developed a light field coder that uses GPU computation for real-time streaming. Our coder is based on a view-dependent coding scheme [Taguchi and Naemura 2006], which represents light field data in a scalable manner. GPGPU (General Purpose Computation on GPUs) techniques as well as conventional rendering functions are incorporated to efficiently perform the view-dependent coding. To enhance compression efficiency for dynamic scenes, our coder uses a simple temporal prediction scheme, which is also implemented on GPU. Using a sequence of integral photography captured with our LIFLET system [Yamamoto et al. 2004], we show experimental results of real-time encoding and decoding.

2 GPU-Oriented Light Field Coder

Our view-dependent coding scheme [Taguchi and Naemura 2006] consists of the following three parts:

(a) Graphical process. A novel image is synthesized with estimating a view-dependent geometry at a certain viewpoint. All of the input multi-view images are then predicted by warping this synthesized image. This corresponds to a spatial prediction.

(b) General process. For each macroblock of 16 $\times$ 16 pixels, a coding mode is assigned depending on the prediction accuracy. This step needs evaluating the quality of each predicted macroblock as PSNR value, and calculating the residual prediction errors.

(c) Compression. The synthesized image and the residual prediction errors (only for the macroblocks where the prediction is not accurate) are compressed using a DCT-based scheme. The view-dependent geometry is also included in the compressed stream.

In these processes, (a) graphical process is suitable for GPU since conventional rendering functions can be used for fast image synthesis. Moreover, (b) general process can be performed efficiently using GPGPU techniques, since it deals with the image data in parallel. Thus our coder reduces the encoding time. These processes are inversely performed in the decoder, in which GPU computation is applied in a similar way. In our current implementation, (c) compression is performed on CPU.

For dynamic scenes, we use a simple temporal prediction scheme to enhance compression efficiency. Only the difference between the current frame and the previous frame is calculated for real-time processing, although motion compensation-based method may achieve higher compression efficiency at the cost of computation complexity. For each macroblock, the spatial or temporal prediction mode is selected depending on the prediction accuracy. This is also computed on GPU.

3 Experiments

We used a sequence of integral photography captured with our LIFLET system to evaluate our coder. An integral photography consists of a set of elemental images, each of which corresponds to a single input view. Figure 1 shows an input integral photography.

We compared three encoding and decoding methods shown in Figure 2. Each method performs (a) graphical process and (b) general process either on CPU or on GPU. Table 1 shows the encoding/decoding frame-rates on a PC (Table 2), using 644 (28$\times$23) elemental images of 31$\times$31 pixels. The temporal prediction was not used in these measurements. These processing times were proportional to the number of elemental images. It can be seen that our coder greatly benefits from the GPU computation for real-time processing.

Using the temporal prediction in the "GPGPU applied" method, the reconstruction quality (PSNR) improved about 6 dB at the same bit rate, while the encoding frame-rate decreased to 14.23 fps. By contrast, the decoding frame-rate increased to 29.40 fps. This is because the residual prediction errors, which should be decoded on CPU, were decreased due to high accuracy of temporal prediction.

4 Conclusions and Future Work

We proposed a GPU-oriented light field coder and achieved real-time encoding and decoding for our LIFLET system. Future work will be focused on handling larger data such as multi-view videos captured with a camera array. It would be important to design a system efficiently distributing the computation to both CPU and GPU.

Acknowledgements: Special thanks go to Prof. H. Harashima and Dr. K. Takahashi at the University of Tokyo.

References

TAGUCHI, Y., AND NAEMURA, T. 2006. View-dependent coding of light fields based on free-viewpoint image synthesis. In *Proc. IEEE Int. Conf. Image Processing (ICIP 2006)*, 509–512.

YAMAMOTO, T., KOJIMA, M., AND NAEMURA, T., 2004. LIFLET: Light field live with thousands of lenslets. ACM SIGGRAPH 2004 Emerging Technologies #0130.

*e-mail:ando@hc.ic.i.u-tokyo.ac.jp