

Real Time 360° Video Stitching and Streaming

Rodrigo M. A. Silva^{1,2*}

Bruno Feijó¹

Pablo B. Gomes²

Thiago Frensh²

Daniel Monteiro²

Pontifical Catholic University of Rio de Janeiro¹
Globo TV Network – Visual Effects Research Group²

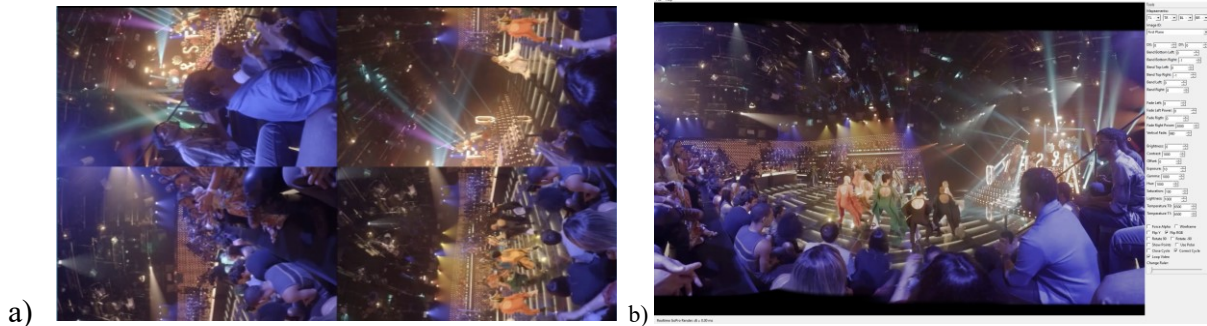


Figure 1: Four camera stitching process. a) Combined input images b) Processing software showing the output

Abstract

In this paper we propose a real time 360° video stitching and streaming processing methodology focused on GPU. The solution creates a scalable solution for large resolutions, such as 4K and 8K per camera, and supports broadcasting solutions with cloud architectures. The methodology uses a group of deformable meshes, processed using OpenGL (GLSL) and the final image combine the inputs using a robust pixel shader. Moreover, the result can be streamed to a cloud service using h.264 encoding with nVenc GPU encoding. Finally, we present some results.

CCS Concepts: • Computing methodologies → Virtual reality;
• Theory of computation → Vector / streaming algorithms

Keywords: virtual reality, video stitching, gpu processing

1 Introduction

The image stitching process is a well know algorithm [Brown and Lowe 2007] for non-real time applications. However, the growing of immersive video production for HMDs (Head Mounted Displays) forces the improvement of these techniques to operate in high performance. The most important field is 360o video production for live events, which imposes a drastic processing time reduction, and requires a rigorous set of deterministic processing, encoding and transmission algorithms.

*e-mail: rodrigo.silva@tv Globo.com.br

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

Copyright is held by the owner/author(s).
SIGGRAPH '16 Posters, July 24-28, 2016, Anaheim, CA,
ACM 978-1-4503-4371-8/16/07.
<http://dx.doi.org/10.1145/2945078.2945148>

In this paper we propose a processing methodology focused on GPU use, which reduces processing time and create a scalable solution for use in large resolutions such as 4K and 8K per camera.

2 Our Approach

The present work uses 4 (figure 1a) or 6 cameras to produce videos with 360° or 180° field of view in real time. In such model, we use three video capture cards (two inputs each) with a Genlock synchronization mechanism. Furthermore, to keep the cameras fixed and provide power and connect the HDMI cables to the capture card, we design and build a camera rig (figure 2a and 2b).

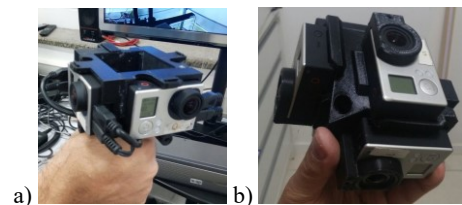


Figure 2: Cameras Rig a) 4 Cameras b) 6 Cameras

Using the Figure 3 as reference, to perform the task, the solution has four processing threads (with shared OpenGL contexts). The first one loads each video frame into video card memory, using a circular buffer. After that, using the synchronization, the system can perform the stitching using the last 4 or 6 frames in the buffer. The stitched image is pushed to another circular buffer and prepares to be encoded using the OpenGL texture ID. Finally, the encoder outputs the byte stream to the CPU and the last thread transmits it over the network

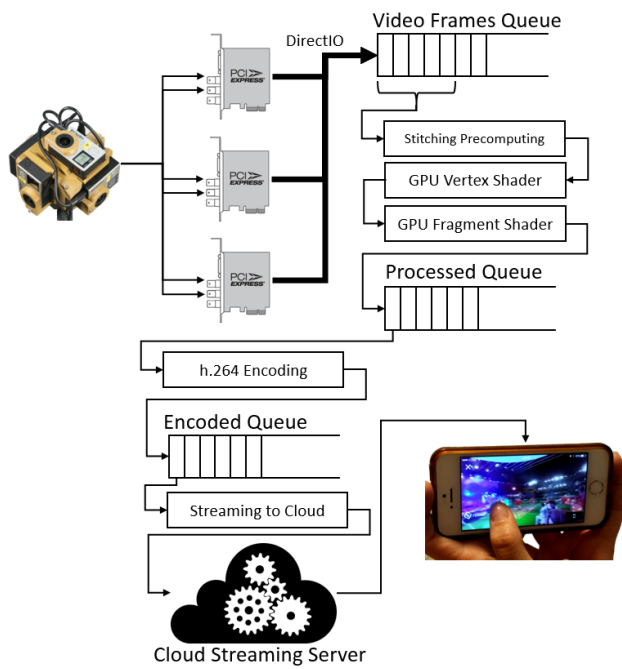


Figure 3: Processing Workflow

3 Processing Methodology

The system concept does not use video specific hardware solutions, however, to perform those tasks it is important to use very specific GPU features and an accurate thread management procedure.

Therefore, the first process makes copies in parallel to GPU, using the PBO (Pixel Object Buffer) interface. The stitching algorithm uses a tessellated plane to map the input images. This planes can be deformed using some pre-defined distortions (radial, linear, manual moving), each one is processed with the vertex shader (figure 4). The solution automatically computes the values using Surf [Bay et al. 2008] and RANSAC [Fischler and Bolles 1981] method (to create the features between the input frames), then it minimizes the input parameters to reduce the distance between features.

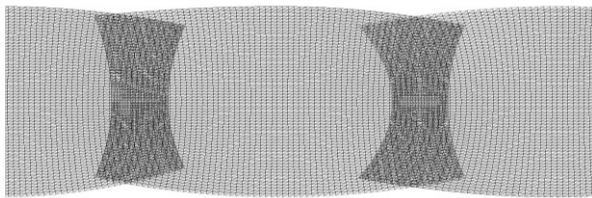


Figure 4: Precomputed Mesh

At runtime, the user can load a pre-calibrated mesh or use interval to compute new values (processed in a parallel thread), also it can manually change the values to accommodate fine details (figure 1b). Lastly, it is processed by a robust fragment shader, that computes the blending and color correction, also manual changes can be made.

Moreover, the result is sent to NVenc Library [NVIDIA 2012] that performs H.264 encoding and copies the final byte stream to

the CPU, after that, the system transmits it to a HMD. The receiver system is a mobile application with a simple H.264 decoder. When it receives the stream and decodes it, the image is projected onto a sphere (or cylinder), creating the sense of immersion (Figure 5).

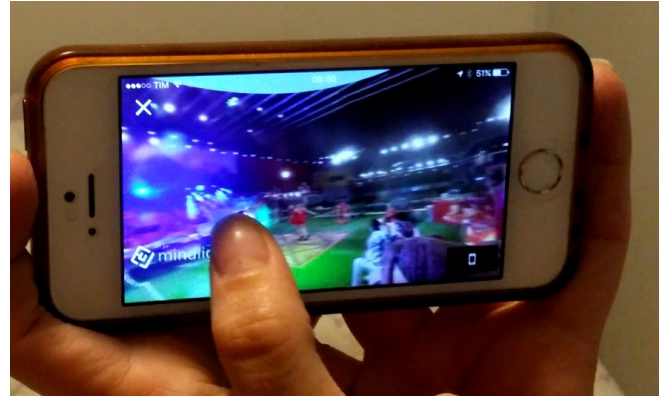


Figure 5: Projected Image onto a cylinder

4 Results

Finally, applying those techniques, the total processing time, after receiving the frame from the video capture card and before transmitting, is about 70 ms, using one video card for processing and 4 video cameras (GoPro Hero 4 Black). The frame size is a Full HD image with 8 bits per channel (RGB). The solution proved to be stable and scalable. Also, some improvements can be developed, such as Pinned Memory for DMA between the video capture card and the GPU (called DirectIO). Moreover, it is possible to perform broadcasting using cloud architectures like Wonza (that uses Amazon EC2).

References

- BROWN, M. AND LOWE, D., 2007. Automatic Panoramic Image Stitching using Invariant Features. *International Journal of Computer Vision*, 74(1), pages 59-73. DOI: <http://dx.doi.org/10.1007/s11263-006-0002-3>
- BAY, H., ESS, A., TUYTELAARS, T., AND GOOL, L. V.. 2008. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*. 110, 3 (June 2008), 346-359. DOI=<http://dx.doi.org/10.1016/j.cviu.2007.09.014>
- FISCHLER, M. A. AND BOLLES, R. C. 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM* 24, 6 (June 1981), 381-395. DOI=<http://dx.doi.org/10.1145/358669.358692>
- NVIDIA. 2012. Nvenc – nvidia kepler hardware video encoder. [Online: <https://developer.nvidia.com/nvidia-video-codec-sdk>]