Dima, E., Sjöström, M., Olsson, R., Kjellqvist, M., Litwic, L. et al. (2018) LIFE: A Flexible Testbed For Light Field Evaluation, in 3DTV-Conference, 3-5 June 2018.

LIFE: A FLEXIBLE TESTBED FOR LIGHT FIELD EVALUATION

Elijs Dima[⋆]*, Mårten Sjöström*[⋆]*, Roger Olsson*[⋆]*, Martin Kjellqvist*[⋆]*, Lukasz Litwic*[†]*, Zhi Zhang*[†]*, Lennart Rasmusson*[‡]*, Lars Flodén*[‡]

[⋆] Dept. of Information Systems and Technologies, Mid Sweden University, Sundsvall, Sweden.
[†] Ericsson Research Digital Representation & Interaction, Ericsson AB, Stockholm, Sweden.
[‡] Observit AB, Sundsvall, Sweden.

ABSTRACT

Recording and imaging the 3D world has led to the use of light fields. Capturing, distributing and presenting light field data is challenging, and requires an evaluation platform. We define a framework for real-time processing, and present the design and implementation of a light field evaluation system. In order to serve as a testbed, the system is designed to be flexible, scalable, and able to model various end-to-end light field systems. This flexibility is achieved by encapsulating processes and devices in discrete framework systems. The modular capture system supports multiple camera types, general-purpose data processing, and streaming to network interfaces. The cloud system allows for parallel transcoding and distribution of streams. The presentation system encapsulates rendering and display specifics. The real-time ability was tested in a latency measurement; the capture and presentation systems process and stream frames within a 40 ms limit.

***Index Terms* —** Multiview, 3DTV, Light field, Distributed surveillance, 360 video

1. INTRODUCTION

Light is the most important medium through which people perceive the 3D world [1]. However, conventional photography and displays can only represent a 2D projection of the 3D world. The effort to record and replicate the world's 3D information through has led to the plenoptic function [2], 3DTV, 360 video, and light fields [3]. Recording, distributing and presenting light field data poses unique challenges in device management and data processing. In order to address these challenges, there is a need for flexible light field test systems on which different solutions for recording, distributing and presenting light fields can be evaluated.

As Wu et al. show [1], the plenoptic and light field research areas contain problems related to capture, processing, distribution and presentation. Capture of spatial and angular light information requires special cameras or camera systems, which impose resolution tradeoffs and uneven sampling rates. The processing and distribution of light fields is complicated by the recorded data quantities, limited bandwidths, lacking format standards, and the use of intermediate data such as depthmaps. The presentation of light fields is also affected by the bandwidth requirements and the variety of image rendering methods and devices.

In this paper, we describe the Light Field Evaluation (LIFE) system: a multi-camera array and data processing and streaming framework designed for investigating the above problems. This framework has been designed to be modular in hardware and software domains, and to serve as a test-bed for light field capture, processing, distribution and presentation. We detail the LIFE system's components and present the 11-camera, 11-computer, 1-cloud prototype implementation as shown in Fig. 1.

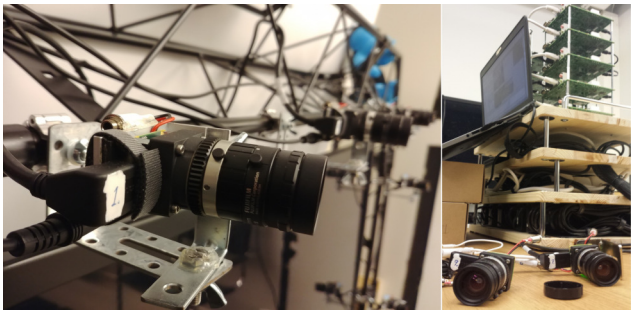

Figure 1. Prototype of the LIFE multi-camera capture array (left) and near-camera processing cluster (right).

2. RELATED WORK

We define *end-to-end light field systems* as systems that include capture, distribution and presentation of 360-degree, 3D or light field data. The required data can be recorded by plenoptic cameras [4] and multi-camera systems [5, 6]. *Plenoptic cameras* produce a scene sampling that closely matches the four-dimensional parametrization of light field [3]. However, plenoptic cameras are seldom used in end-to-end systems, due to the spatial-angular resolution tradeoff and high data bandwidth [1]. *Multi-camera systems* avoid the bandwidth limits of plenoptic cameras, and therefore are used in end-to-end systems. Such systems have variously been presented as 3D-TV [7] or light field [5, 6] systems, depending on the choice of data format and application.

The 3D-TV system in [7] uses 16 cameras controlled by 8 computers, which compress the video using MPEG-2. The compressed data is sent to a cluster of rendering computers which decode and provide frame data to the renderer. The renderer is described as either a direct per-camera input to a projector, or as a linear pixel value interpolation process that uses a lookup table of input-to-output pixel mapping. The mapping is based on a pre-capture calibration and view selection. Cameras are hardware-synchronized. However, the system does not allow to vary the camera positioning, does not support multiple camera types, and does not allow any data processing besides encoding, decoding, and lookup-table based rendering.

The light field system by Yang et al. [5] uses six computers to receive data from 64 cameras. A seventh computer is tasked with rendering the captured light field to a 2D display. The rendering algorithm projects sections of images from all cameras onto a pre-specified focal plane, and renders the output image by focal plane projection. Calibration is achieved off-line, and synchronization is based on computer clock equalization. Bandwidth is limited by sending image parts which project into the output image. However, the system lacks the possibility to have a sparse camera spacing, to vary the camera layout, and to transfer the en-

978-1-5386-6125-3/18/$31.00

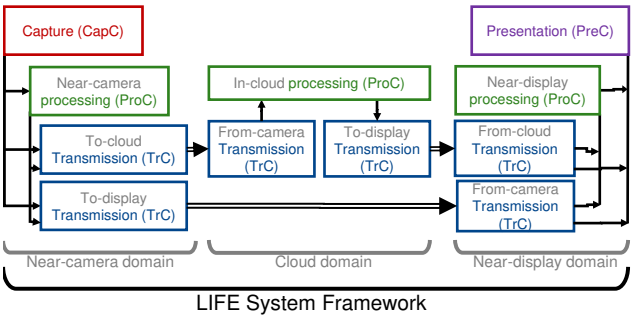

Figure 2. High-level framework of the LIFE system. Capture (red), processing (green), transmission (blue) and presentation (purple) components are distributed in three domains with high internal bandwidth. Arrows indicate flow of recorded data.

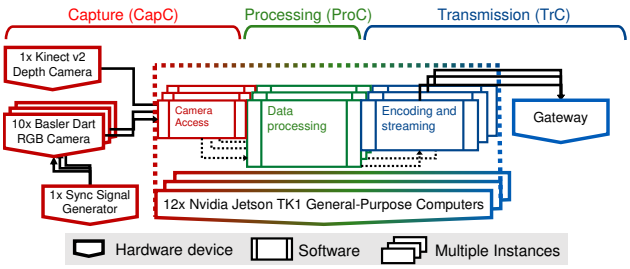

Figure 3. Implementation of near-camera domain of the LIFE framework. Arrows show main data flow.

tire light field to the renderer and display devices.

Balogh et al. [6] use 27 2D cameras controlled by a single computer. This computer is directly connected to a 3-computer cluster that drives a micro-projector based display. Bandwidth is reduced by in-camera hardware Motion-JPEG compression. In the computing cluster, each camera has a dedicated CPU thread for decompression. The decompressed data is sent to a GPU for rendering. The rendering consists of mapping the recorded light rays from cameras to the emitted light rays on the projector display. Camera calibration is done off-line, and synchronization is based on computer clock equalization. However, the system lacks the possibility to vary camera type and layout, to synchronize the acquisition, and to support several display technologies.

All of these systems stick to homogeneous arrays of off-the-shelf cameras. All use rendering methods that are tuned for the display technology, and rely on pixel-to-pixel mapping. Connection between capture and presentation devices is direct. Scalability is achieved by copying the existing components. The design does not consider processing the recorded data or modifying hardware and software components.

3. LIGHT FIELD EVALUATION SYSTEM FRAMEWORK

In this section we describe the flexible high-level framework of the LIFE system. The framework is based on three domains: *near-camera*, *cloud* and *near-display domain*, as shown in Fig. 2. Each domain is an environment with low-delay, large-bandwidth communication. Across domain borders, communication has less bandwidth and more latency. Each domain is distinguished by containing cameras, cloud computers, or presentation displays. Within the domains, devices and processes are grouped in four components: Capture, Processing, Transmission, and Presentation. Fig. 2 shows the component and domain grouping.

The component-based design of the LIFE framework achieves two important goals: ability to model diverse light field systems, and system flexibility, which are key requirements for a light field testbed. System flexibility ensures that these configurations can be implemented, and that changes in one component do not require a full system reimplementation.

The Capture Component (CapC) in Fig. 2 consists of the system's cameras and camera control devices. The CapC is defined only in the *near-camera domain*. Camera interaction processes, including synchronization, image acquisition and camera control, are contained in the CapC. The CapC is responsible for supplying recorded data to the processing and transmission components using non-camera-specific formats. By abstracting the camera interaction, the LIFE system becomes flexible with respect to new camera hardware, because camera changes only affect the CapC. This abstraction also ensures that camera choice does not force a specific distribution and rendering model.

The Processing Component (ProC) contains processes for two kinds of operations: ones that generate new information from the recorded data, and ones that significantly alter the existing information. The ProC is defined and found in all three domains of the LIFE system. In the *near-camera domain*, the ProC can take advantage of the low-latency availability of the data. In the *cloud domain*, the ProC can make use of computational resources of networked computer clusters, typically unavailable in near-camera or near-display domains. In the *near-display domain*, the ProC can provide the last-step computation required for rendering and presentation. Distributing the processing (e.g. calibration, depth estimation, transcoding) between the domains allows the LIFE framework to model and implement a variety of light field systems.

The Transmission Component (TrC) consists of the data distribution mechanisms, including networking, data stream forming, compression and decompression. The TrC is responsible for data flow between domains via links with possible bandwidth and latency constraints. In this way, the TrC allows the LIFE system to support and model various distribution methods and technologies.

The Presentation Component (PreC) consists of the rendering process, display hardware, and display control processes and the display, and so is only defined in the *near-display domain*. The PrC includes the rendering process, because in practice the display technology tends to strictly enforce the use of a specific rendering approach. This abstraction of display technology allows the LIFE system to have non-display-specific implementations, and scale from 2D, via stereoscopic, to light field presentation without significant changes in the preceding components.

4. IMPLEMENTATION

In this section we describe the implementation of the LIFE system components.

4.1. Near-Camera Domain

The near-camera domain is shown in Fig. 1 and 3. The implementation is made up of functionally parallel units, each with one camera and one computer. This unit-based approach improves upon the light field systems mentioned in section 2, by allocating more processing resources and bandwidth per camera. As a result, the LIFE system can be used to try out concepts for smart-camera systems used in surveillance to provide in-camera processing. As shown in Fig. 3, the cameras are the hardware part of CapC, and the computers are the hardware platform for running the CapC, ProC and TrC software.

The hardware is based on camera-computer units. Each unit consists of a General-Purpose Computer (GPC) and a camera de-

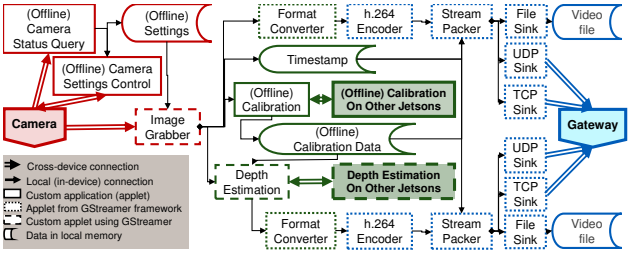

Figure 4. Communication between LIFE and GStreamer components on a Jetson TK1 in the near-camera domain. Arrows show data flow. "Offline" means component does not execute during video streaming. Shaded background means component is on another device.

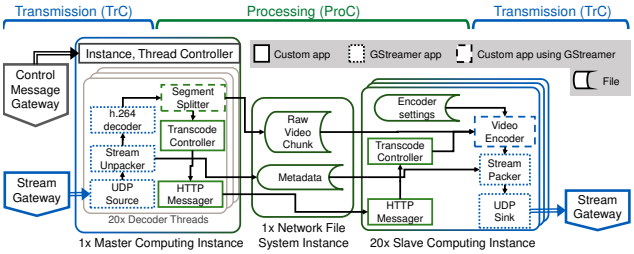

Figure 5. Cloud domain prototype implementation. Instances are virtual computers in OpenStack based cloud platform. Gateways are entry and exit points to the cloud platform. Arrows show data flow.

vice. We use the NVidia Jetson TK1 computers as GPCs, because they have a range of high-bandwidth interfaces, a multi-core CPU and GPU, and a Linux Ubuntu operating system with full support for generic applications. We capture light position and angle with ten Basler daA1600-60uc cameras, and one Kinect v2 is used for 2D and depth sensing. The Basler cameras have a small form factor, allowing wide-baseline and narrow-baseline configurations for parallel, converged and diverged view directions. The implementation supports both hardware and software synchronization. This enables to simulate different applications and the consequence synchronization errors have on data consistency [8]. We use a signal generator to provide a synchronization signal to the Basler cameras, and the GPCs are software-synchronized using Network Time Protocol. The GPCs are connected to each other by a local GigE network, with a gateway to the Internet to stream data to the cloud and to a received with a display.

Software implements the Capture, Processing and Transmission components in application modules (applets) on each GPC, see Fig. 4. The implementation merges custom LIFE applets with standard applets in the GStreamer [9] framework. The four main processes are *image streaming*, *depth streaming*, *camera calibration* and *configuration*. *Calibration* and *configuration* are offline processes, disabled during video recording. Output from these processes is stored in local files, which are accessed by other applets during recording. The offline, target-based calibration method is chosen via assessments in [10].

The *image streaming* pipeline applets in Fig. 4 are image grabber, format converter, H.264 encoder, stream packer, and sinks. The image grabber uses camera-specific API, saves a timestamp for each frame, and provides the frame through a GStreamer source element. Each recorded frame is converted to the input format of the GStreamer-based H.264 encoder. The stream packer appends the recorded frame, timestamp and camera calibration metadata to the videostream. GStreamer sinks are used to send the videostream across the local network or to local storage. The *depth streaming* pipeline splits from the image streaming pipeline after the image grabber applet. In a multiview-plus-depth case, depth data is generated by the depth estimation applet in the depth streaming pipeline. The depth estimation applet is simulated by image filtering to a 16-bit grayscale matrix.

4.2. Cloud Domain

A private Ericsson Research data center is used to implement the LIFE cloud domain. The data center serves as a cloud research laboratory, and provides a compute platform for data analytics and simulations. The data center is equipped with 1000+ servers and 100+ GPUs, connected via 100-Gigabit network into an OpenStack cloud service. The prototype cloud is used to transcode high-bitrate video streams from the near-camera domain to standard bitrate streams for end consumers. Other uses include distributed depth estimation, view rendering, and end-to-end delivery optimization. The prototype implementation is based on virtual computer instances, and uses a master instance, a file-system data cache instance, and a pool of slave instances to process video streams as shown in Fig. 5.

The master instance acts as the cloud controller and the endpoint for video streams inbound from near-camera domain. A TCP-based service is constantly listening for HTTP-request based instructions for stream configuration, monitoring and launching. Upon stream launch, the master instance sets up the requested number of GStreamer stream decoder threads as shown in Fig. 5, and starts a slave instance for each stream. Slave instances are directly controlled by master instance via HTTP requests. During streaming, incoming video data is decoded on the master instance, and stream metadata and raw frames and are are cached to the Network File System Cache (NFSC). The NFSC allows to bypass network protocol overheads and reduces the master-to-slave data transmission time.

Each slave instance is listening to file system events on the NFSC. When a raw video frame is available in the assigned stream, the slave instance uses local encoder settings and configuration from the master to re-encode the video. Due to the computing resource availability, the re-encoded stream can have a better quality to compression ratio and different data format. The encoded video is re-packed into a Real-Time-Protocol stream with the original metadata, and sent through a dedicated exit gateway. The transcoding process on each slave instance can be stopped and restarted at will. When re-started, the process returns to encoding and streaming the newest frame in the NFSC, in order to allow the restarted slave instance to keep up with other active slave instances. The parallel streaming architecture allows each outgoing stream to have a different bitrate and encoding quality. Moreover, per-stream processing tasks can be deployed on slave instances without affecting stream management and transcoding organization.

4.3. Near-Display Domain

The near-display domain is implemented on a generic computer, connected to an autostereoscopic display seen in Fig. 1. Transmission is handled by the GStreamer network source components and video decoder. A GStreamer video sink element serves as the prototype presentation component. It is foreseen to further develop the near-display domain in the future.

5. PRELIMINARY RESULTS

The Basler cameras generate YUV422 video at 25 Frames Per Second (FPS), at the maximum sensor resolution of 1600 by 1200

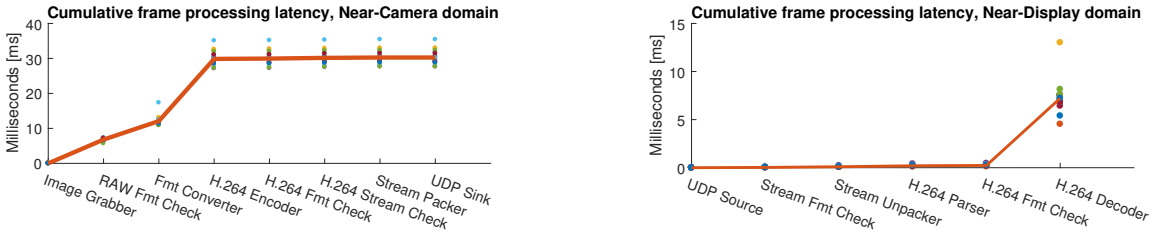

Figure 6. Streamed video frame processing latency in our system's near-camera and near-display domains. Displayed latency is cumulative. Frames are processed by the modules listed on x-axes. Red line shows average frame latency, dots show individual frame latency observations.

pixels. We measured the cumulative latency for frame processing in near-camera and near-display domains, to verify that the framework is able to process and transmit the video data. Domains were connected via a public network, in order to create realistic conditions for the transmission components. The results shown in Fig. 6 demonstrate that both near-camera and near-display domains can operate within the real-time requirement of 40 ms, set by the 25 FPS recording rate. The cameras take 9.5 ms to expose the sensor, and 20 ms to read out and debayer the recorded frame at full resolution. Because this process is only camera-dependent, we do not consider the in-camera exposure and debayering time as framework processing latency.

In near-camera domain, the H.264 encoder runs on the GPU and the remaining components occupy one CPU core, leaving three cores available for the framework's processing modules such as depth estimation and depth streaming. The significant sources of latency are the H.264 encoder and the frame format converter, used to change from interleaved (YUYV) to planar (I420) frame format. The H.264 encoder latency includes moving data between CPU and GPU memory. In the near-display domain, the only notable latency is incurred by the H.264 decoder. In both domains, network control overheads are negligible.

6. CONCLUSIONS AND FUTURE WORK

We have presented a flexible Light Field Evaluation framework, and an implementation covering the framework domains. The framework is designed to allow modelling various light field end-to-end systems, by classifying components into near-camera, cloud, and near-display domains. Processing of recorded light field data can be located on any domain.

The near-camera domain prototype has been implemented as a model of a smart camera system. Every camera is paired to a general-purpose computation device, able to process, encode and stream the recorded data to the cloud domain or presentation devices. This design allows to support various camera types and avoid near-camera bandwidth restrictions. Small camera form and flexible mounts support various multi-camera arrangements. The near-camera prototype has been shown to handle recorded frame processing and sending within real-time constraints imposed by the recording camera framerate, at full sensor resolution. The cloud domain prototype has been implemented in an Ericsson Research data center as a scalable master-slave architecture. The cloud platform gives access to massively-parallel computing resources for light field processing and video stream transcoding.

Future work is focused on developing the processing components of the framework, such as near-camera depth estimation and cloud-assisted light field rendering. Various cameras and display technologies will be included as supported modules within the end-to-end system. The system will be used as a testbed for research and evaluation of light field and multiview capture, processing, distribution and presentation.

7. ACKNOWLEDGMENT

This work has been supported by the LIFE project grant 20140200 of the Knowledge Foundation, Sweden.

8. REFERENCES

[1] Gaochang Wu, Belen Masia, Adrian Jarabo, Yuchen Zhang, Liangyong Wang, Qionghai Dai, Tianyou Chai, and Yebin Liu, "Light field image processing: An overview," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 7, pp. 926–954, 2017.

[2] Edward H Adelson and James R Bergen, *The Plenoptic Function and the Elements of Early Vision*, pp. 3–20, Vision and Modeling Group, Media Laboratory, Massachusetts Institute of Technology, 1991.

[3] Marc Levoy and Pat Hanrahan, "Light field rendering," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1996, pp. 31–42.

[4] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan, "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report CSTR*, vol. 2, no. 11, pp. 1–11, 2005.

[5] Jason C Yang, Matthew Everett, Chris Buehler, and Leonard McMillan, "A real-time distributed light field camera.," *Rendering Techniques*, vol. 2002, pp. 77–86, 2002.

[6] Tibor Balogh and Péter Tamás Kovács, "Real-time 3D light field transmission," in *Real-Time Image and Video Processing*. International Society for Optics and Photonics, 2010, vol. 7724, p. 772406.

[7] Wojciech Matusik and Hanspeter Pfister, "3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes," in *ACM Transactions on Graphics (TOG)*. ACM, 2004, vol. 23, pp. 814–824.

[8] Elijs Dima, Mårten Sjöström, and Roger Olsson, "Modeling depth uncertainty of desynchronized multi-camera systems," in *2017 International Conference on 3D Immersion (IC3D)*. IEEE, 2017, pp. 1–6.

[9] GStreamer Developers, "GStreamer: open source multimedia framework," 2018, https://gstreamer.freedesktop.org/.

[10] Elijs Dima, Mårten Sjöström, and Roger Olsson, "Assessment of multi-camera calibration algorithms for two-dimensional camera arrays relative to ground truth position and direction," in *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*. IEEE, 2016, pp. 1–4.