A REAL-TIME COARSE-TO-FINE MULTIVIEW CAPTURE SYSTEM FOR ALL-IN-FOCUS RENDERING ON A LIGHT-FIELD DISPLAY

Fabio Marton, Enrico Gobbetti, Fabio Bettio, José Antonio Iglesias Guitián and Ruggero Pintus

CRS4 Visual Computing Group
http://www.crs4.it/vic/

ABSTRACT

We present an end-to-end system capable of real-time capturing and displaying with full horizontal parallax high-quality 3D video contents on a cluster-driven multiprojector light-field display. The capture component is an array of low-cost USB cameras connected to a single PC. Raw M-JPEG data coming from the software-synchronized cameras are multicast over Gigabit Ethernet to the back-end nodes of the rendering cluster, where they are decompressed and rendered. For all-in-focus rendering, view-dependent depth is estimated on the GPU using a customized multiview space-sweeping approach based on fast Census-based area matching implemented in CUDA. Real-time performance is demonstrated on a system with 18 VGA cameras and 72 SVGA rendering projectors.

***Index Terms*—** Multi-view capture and display, GPU, light field rendering

1 INTRODUCTION

Our system aims at real-time capturing 3D scenes using an array of cameras while simultaneously displaying them on a remote cluster-driven multi-projector 3D display able to deliver 3D images featuring continuous horizontal parallax to multiple naked-eye freely moving viewers in a room-sized workspace.

Many camera array systems (and fewer multi-projector 3D display systems) have been presented in the past for acquiring and displaying 3D imagery, covering all the spectrum from pure image-based representations to full geometric reconstruction. In this paper, we describe a real-time system constructed around an on-the-fly coarse-to-fine depth estimation method which synthesizes an all-in-focus light-field representation on the display side by finding the optimal depth value for each pixel of the 3D display. The rendering algorithm is fully implemented on a GPU using GPGPU techniques.

2 RELATED WORK

Our system extends and combines state-of-the-art results in a number of technological areas. In the following, we only discuss the approaches most closely related to ours. We refer the reader to established surveys (e.g., [1, 2]) for more details.

Multi-camera capture and display. A number of papers describing real-time 3D video or light-field capture and display have been published in recent years, achieving significant advances. One of the approaches consists in using a pure light field method, in which images captured by source cameras are regarded as sets of rays sampled from the camera's position, and images are rendered by re-sampling from the database the rays which pass through the rendering viewpoint (e.g., [3, 4, 5]). Little processing is required, and the quality of the rendered image is potentially very high in terms of photo-realism, However, accordingly with plenoptic sampling theory [6], the scene is adequately imaged only relatively close to the focal plane if one needs to cover a wide field of view with not too-many cameras, which makes pure light field systems not fully scalable. Using geometric information, which for real-time systems must be estimated on-the-fly, it is possible to create higher quality views with less cameras. Since globally consistent models are hard to construct within strict time budgets, real-time systems for general scenes are based on view-dependent approximate depth reconstruction [7, 8, 9, 10]. These methods exhaustively evaluate depth hypotheses for each pixel, which makes them prone to local minima during correspondence search, and reduces their effective applicability to high pixel count displays. In this work, we extend a coarse-to-fine stereo-matching method [11] to real-time multiview depth estimation using a space-sweeping approach and a fast Census-based [12] area matching, and integrate it in a rendering system for the peculiar multi-projector 3D display imaging geometry. We also describe a full end-to-end implementation achieving real-time performance using commodity components.

Rendering for multi-projector light field display. The display hardware employed in this work has been developed by Holografika[1] and is commercially available. Our image generation methods take into account the display characteristics in terms of both geometry and resolution of the reproduced light fields. In particular, we extend a multiple-center-of-projection technique [13, 14] with a depth compression factor, and use the display geometry within the space-sweeping step. We use the common sort-first parallel rendering approach, multicasting all images to rendering nodes for depth reconstruction and light field sampling. The assignment between rendering processes and images is static, even though load balancing strategies based on image partitioning could be implemented on top of our framework (e.g., [15]).

3 SYSTEM OVERVIEW

Our system acquires a stream video as a sequence of images from a camera array and reconstructs in real-time the 3D scene on a light field display with full horizontal parallax. The display used in this work filters through a holographics screen the light coming from specially arranged array of projectors controlled by a PC cluster (see Fig. 1). The projectors are densely arranged at a fixed constant distance from a curved (cylindrical section) screen. All of them project their specific image onto the holographic screen to build up a light field. Mirrors positioned at the side of the display reflect back onto the screen the light beams that would otherwise be lost, thus creating virtual projectors that increase the display field of view. The screen has a holographically recorded, randomized surface relief structure

978-1-61284-162-5/11/$26.00 © 2011 IEEE

[1] www.holografika.com

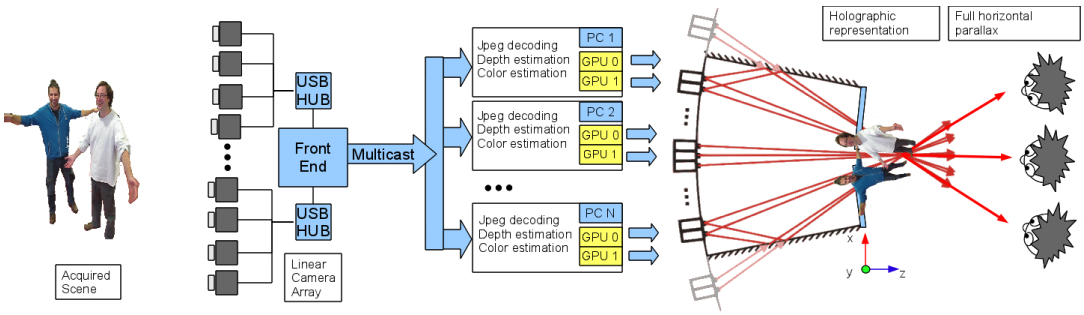

Fig. 1: ***Overall system concept.*** *A linear camera array is connected through USB 2.0 to a front-end PC which captures the 3D scene in M-JPEG format. Each frame is packed and multicast to rendering PCs, which perform JPEG decoding, per-view depth estimation, and light field sampling to produce projector images for the light field display.*

able to provide controlled angular light divergence: horizontally, the surface is sharply transmissive, to maintain a sub-degree separation between views determined by the beam angular size Φ. Vertically, the screen scatters widely, hence the projected image can be viewed from essentially any height. With this approach, a display with only horizontal parallax is obtained.

A master PC performs image acquisition in JPEG format from a linear camera array connected to a single capturing PC through USB 2.0. When using M-JPEG, up to 9 cameras at capturing $640 \times 480@15Hz$ can be connected to a single USB port. Cameras are software synchronized, and all images of a single multiview frame are assembled and distributed to light-field rendering clients through a frame-based reliable UDP multicast protocol. Each rendering node manages a small group of projectors, and at each frame decodes JPEG images to a 3D RGB array directly on the GPU and produces an all-in-focus 3D image by casting projector rays, estimating scene depth along them with a coarse-to-fine multiview method, and resampling the light field using a narrow aperture filter.

4 CALIBRATION

Our system assumes that both the input camera array and the display projectors are calibrated in both intrinsic and extrinsic parameters.

The camera array is calibrated by first applying Tsai's method [16] using images of a checkerboard positioned at various location within the camera workspace, and then globally refining all camera parameters with a bundle adjustment step [17].

For the 3D display, we derive geometric calibration data by suitably modifying existing automated multi-projector calibration techniques [18]. We build on the classic two-step approach in which position and frustum of each projector are found through parametric optimization of an idealized pinhole model and any remaining error is corrected using a post-rendering 2D image warp that "moves" the pixels to the correct idealized position. For the first calibration step, we assume that projector positions are known, and we find their orientation and projection matrices using vision based techniques. The whole calibration procedure is performed by covering the holographic surface with a standard white diffuser and photographing it from inside. For each projector, we first project an asymmetric pattern and analyze projected images to determine mirror positions (see Fig. 2 left). If no mirror is detected, we associate a single viewport to the projector; otherwise, we create a "virtual projector" on the other side of the mirror and proceed with calibration by separately handling the mirrored and direct view rendered in different viewports. In this calibration step, a checkerboard pattern is projected onto the screen (see Fig. 2 center and right). The full orientation and the perspective matrix are derived from the detected corners. The remaining errors are then corrected using a cubic polynomial approximation method for mapping undistorted to distorted coordinates.

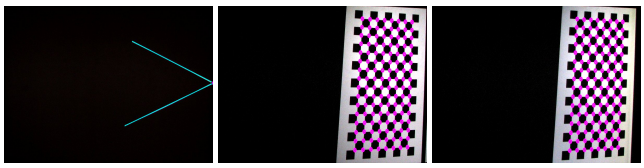

Fig. 2: ***Light field display calibration.*** *An asymmetric pattern detects a mirror. Checkerboard patterns are then projected in the direct and mirror image to calibrate the projector.*

5 REAL-TIME ALL-IN-FOCUS RENDERING

In our cluster-parallel implementation, we run one separate multithreaded back-end rendering process per node, and connect each of them to a coordinator process running on the front-end PC using MPI for commands and using multicast for image data. Each back-end process controls a portion of the framebuffer and a separate rendering is performed for each projector view Each time a new multiview frame (consisting in a block of JPEG-compressed images) arrives on the multicast connection, all-in-focus rendering is performed on each projector view at the same resolution of the input images.

First, we perform a pipelined CPU-GPU parallel decoding for the set of images, interleaving the entropy decoding on the CPU using libjpeg-turbo [2], with the dequantization, inverse-DCT, and YCbCr to RGB conversion on the GPU by CUDA kernels. Decompressed RGBA images are then stored into a 3D array, one image per slice. All the remaining steps are fully performed on the GPU as a sequence of CUDA kernel invocations.

From the image blocks, we first produce a Gaussian RGBA pyramid for each of the images, constructed with 2D separable convolution of a filter of width 5 and factor-of-two sub-sampling. We then construct a parallel pyramid containing descriptors for each pixel, which will be used in the block matching steps. Each descriptor uses 32 bits, with 24 bits devoted to the Census representation [19] of the 5x5 block centered at the pixel, and the remaining 8 bits used for a luminance value (averaged on a 3x3 block). The Census stores a bit set to one for each neighbor pixel with value higher than pixel value, and zero otherwise. The descriptor allows us to rapidly estimate the similarity between two blocks by weighting the Hamming distance between Census representations, highly informative on textured areas or boundaries, with the absolute difference among the luminance components, which takes care of differentiating flat areas of different colors. Hamming distances are quickly computed in CUDA using the intrinsic population count function `__popc`.

[2] `http://libjpeg-turbo.virtualgl.org`

Through the auxiliary pyramids, we then perform depth estimation using a coarse-to-fine approach, successively refining depth estimates found at each pyramid level. The procedure works by following the ray associated to each 3D display pixel, put it in correspondence with the camera space using a customizable modeling transform. Following [13, 14, 20], we apply a multiple-center-of-projection technique for generating images with good stereo and motion parallax cues. The technique is based on the approach of fixing the viewer's height and distance from the screen to those of a virtual observer in order to cope with the horizontal parallax only design (see Fig. 3). We assume that the screen is centered at the origin with the y axis in the vertical direction, the x axis pointing to the right, and the z axis pointing out of the screen. Given a virtual observer at $\mathbf{V}$, the ray origin passing through a display screen point $\mathbf{Q}$ is then determined by $\mathbf{O} = ((1-\eta)(V_x)+\eta(E_x+\frac{Q_x-E_x}{Q_z-E_z}(V_z-E_z)), V_y, V_z))$ where $\mathbf{E}$ is the position of the currently considered projector, and η is a interpolation factor, which allows us to smoothly transition from standard single view perspective rendering (with $\eta = 0$) to full horizontal parallax rendering (with $\eta = 1$).

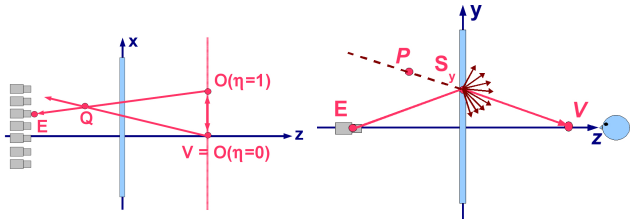

Fig. 3: ***Light field geometry****: Left: horizontally, the screen is sharply transmissive and maintains separation between views. We smoothly transition from standard single view perspective rendering (with h = 0) to full horizontal parallax (with h = 1). Right: vertically, the screen scatters widely so the projected image can be viewed from essentially any height.*

For coarse to fine depth estimation, we analyze a space bounded by two planes in camera coordinates, and use a function that remaps an index t to a depth z in camera space such that there is decreasing density moving from near plane z_n to far plane z_f: $z = (\frac{1-t}{Z_n} + \frac{t}{Z_f})^{-1}$. At the first step, we initialize a buffer of unknown depths, half the resolution of the coarsest pyramid level, to the index of the center of the depth range. This depth map is up-sampled by a nearest neighbor filter and then used as the initial guess for estimating the depths at the next finer level of the pyramid using only a small fixed number of steps (5 in this paper) and a search range that is half of the whole depth range. This upsampling and estimation process is repeated at successive finer levels, halving the search range at each step but keeping the number of steps constant, until we get the depth map at the finest level.

Depth estimation first independently chooses for each pixel the depth that results in the best matching score among the 4 cameras nearest to the ray. Using only 4 cameras improves scalability, as it makes the matching process independent from the camera count, and reduces problems due to occlusions. The multiview matching score is obtained by computing the intersection point between the display ray and the current camera depth plane, reprojecting the point onto the camera image plane, and averaging the pairwise matching scores among descriptors centered at the reprojected pixel in each camera image. The best depth and costs are then stored in a buffer, on which a 3x3 median cost filter is applied in order to remove clear outliers. Then, a separable 5x5 filter is applied to select the depth with the best matching score in the neighborhood. This filtering step, whose width matches that of the area-based descriptors, has the effect of efficiently matching with non-centered windows near occlusion boundaries, and to try out more depth hypotheses from the coarser level depth map [11] (see Fig 4).

After the process is repeated for all pyramid levels, we obtain an estimate of the depth at each pixel in the rendering viewport. A final CUDA kernel then samples the light field from this depth using a narrow aperture filter, which combines the colors of the 4 nearest cameras with a narrow Gaussian weight. Such a filter allows us to reproduce view-dependent effects such as reflections.

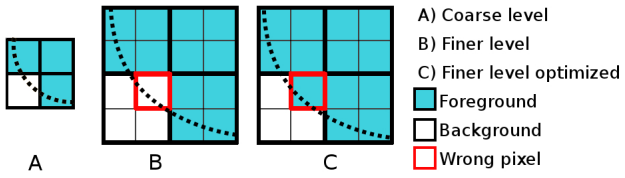

Fig. 4: ***Coarse to fine optimization.*** *Dotted lines represent surface boundary. Left: coarse level estimated depths. Center: refined depths with wrong estimation for red highlighted pixel. Right: corrected pixel depth considering neighbors depths.*

6 RESULTS

Our system has been implemented on Linux using NVIDIA CUDA 3.2. The front-end node is a single Intel Core Quad CPU 2.83GHz Linux PC that is connected through 2 USB hubs to a linear camera array made of 18 cameras. Cameras are Logitech Portable Webcam C905 working at 640×480 @ 15 fps, equally spaced at about 9cm intervals, and calibrated at sub-pixel precision. The captured scene covers a wide field of view and depth (from 1 to 5m). Our 3D display is capable of visualizing 35Mpixels by composing images generated by 72 SVGA LED commodity projectors illuminating a 160×90cm holographic screen. The display provides continuous horizontal parallax within a 50° horizontal field-of-view with 0.8° angular accuracy. The pixel size on the screen surface is 1.5mm. The rendering back-end is currently running on an array of 18 Athlon64 3300+ Linux PCs equipped with two NVIDIA 8800GTS 640MB (G80 GPU) graphics boards running in twin-view mode. Each back-end PC has thus to generate $4 \times 800 \times 600$ pixels using two OpenGL graphics boards with CUDA capability 1.0 and based on an old G80 chip. Front-end and back-end nodes are connected through Gigabit Ethernet. The original M-JPEG images are multicast to the rendering nodes on a single gigabit Ethernet channel with approximately 10% link utilization.

Fig. 5: ***Live capture.*** *Two representative frames recorded using a hand-held video camera freely moving in the camera and display workspace.*

It is obviously impossible to fully convey the impression provided by our end-to-end 3D system on paper or video. As a simple illustration of our system's current status and capabilities, we recorded the performance using a hand-held video camera freely moving in the camera and display workspace [3]. Representative video frames are shown in Fig. 5. Our method appears to reasonably synthesize novel views. A very demanding situation is shown in Fig. 6 right. Although some estimated depth values are incorrect in textureless regions and near boundaries, the rendered colors mostly look visually correct. Small

[3] The video is available from `http://vic.crs4.it/multimedia/`

artifacts can be noticed at some depth boundaries, since our acceleration method based on precomputed pyramids trades quality with speed over a full correlation approach, such as the one employed in adaptive coarse-to-fine stereo [11], which is however hard to apply in real-time to a multiview setting. Images are in all cases sharper than with a pure light field approach with the same camera configuration (see Fig. 6 left).

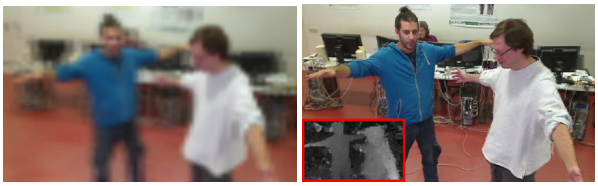

Fig. 6: ***All-in-focus performance.*** *The light-field image (left) is blurry since the camera spacing is insufficient to adequately sample the ray space. By estimating the depth (right), we can produce all-in-focus images. Even though depth estimation may fail in textureless areas, the failures are not visible in the color image.*

Even though the rendering hardware is definitely not high end, the application remained interactive with an average frame rate during the recorded interaction sequence of 9fps. With the same settings, but replacing the graphics boards with a recent medium end one (NVIDIA GTX 460), the unmodified code is capable of reaching the performance of 21fps, therefore exceeding capture performance speed. As shown in Fig. 7, on 8800 GTS GPUs most of the time is spent for data uploading and JPEG decompression, which emphasizes the performance of our all-in-focus renderer. An interesting plan for future work is to exploit the availability of hardware accelerated video codecs on recent graphics boards to reduce overall bandwidth and improve performance.

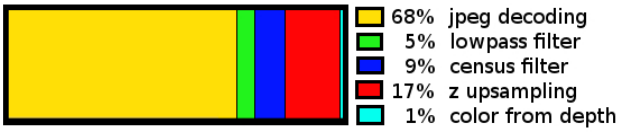

Fig. 7: ***Kernel time plot.*** *Kernel time distribution in a single frame. Time also include memcopy operations, from host to device for JPEG decoding, and from device to device for moving results to cudaArray for accessing them with a texture sampler.*

7 CONCLUSIONS

We have presented an end-to-end system capable of real-time capturing and displaying with full horizontal parallax high-quality 3D video contents on a cluster-driven multiprojector light-field display. Our GPGPU implementation achieves interactive performance on decompression and all-in-focus rendering on CUDA 1.0 graphics boards, and its scalable performance is demonstrated for future generation boards. Our future plans include upgrading the implementation to work on a smaller cluster with recent generation boards, and research on methods for non-linearly remapping the depth range of the captured scene within the display workspace.

8 References

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multiview imaging and 3DTV," *IEEE Signal Processing Magazine*, vol. 24, no. 6, pp. 10, 2007.

[2] A. Smolic, "3D video and free viewpoint video – from capture to display," *Pattern Recognition*, 2010, In press.

[3] W. Matusik and H. Pfister, "3D TV: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 814–824, 2004.

[4] R. Yang, X. Huang, S. Li, and C. Jaynes, "Toward the light field display: Autostereoscopic rendering via a cluster of projectors," *IEEE Trans. Vis. Comput. Graph.*, vol. 14, pp. 84–96, 2008.

[5] T. Balogh and P. Kovács, "Real-time 3D light field transmission," in *SPIE*, 2010, vol. 7724, p. 5.

[6] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, "Plenoptic sampling," in *Proc. SIGGRAPH*, 2000, pp. 307–318.

[7] R. Yang, G. Welch, and G. Bishop, "Real-time consensus-based scene reconstruction using commodity graphics hardware," in *Proc. Pac. Graph.*, 2002, pp. 225–.

[8] Y. Kunita, M. Ueno, and K. Tanaka, "Layered probability maps: basic framework and prototype system," in *Proc. VRST*, 2006, pp. 181–188.

[9] Y. Taguchi, K. Takahashi, and T. Naemura, "Real-time all-in-focus video-based rendering using a network camera array," in *Proc. 3DPVT*, 2008, pp. 241–244.

[10] Y. Taguchi, T. Koike, K. Takahashi, and T. Naemura, "TransCAIP: A live 3D TV system using a camera array and an integral photography display with interactive control of viewing parameters," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, pp. 841–852, 2009.

[11] M. Sizintsev, S. Kuthirummaly, S. Samarasekeray, R. Kumary, H. S. Sawhneyy, and A. Chaudhryy, "Gpu accelerated realtime stereo for augmented reality," in *Proc. 3DPVT*, 2010.

[12] M. Weber, M. Humenberger, and W. Kubinger, "A very fast census-based stereo matching implementation on a graphics processing unit," in *Proc. ICCV Workshops*, 2009, pp. 786 –793.

[13] A. Jones, I. McDowall, H. Yamada, M. T. Bolas, and P. E. Debevec, "Rendering for an interactive 360 degree light field display," *ACM Trans. Graph*, vol. 26, no. 3, pp. 40, 2007.

[14] M. Agus, E. Gobbetti, J. A. I. Guitián, F. Marton, and G. Pintore, "GPU accelerated direct volume rendering on an interactive light field display," *Computer Graphics Forum*, vol. 27, no. 2, pp. 231–240, 2008.

[15] K. Niski and J. D. Cohen, "Tile-based level of detail for the parallel age," *IEEE Trans. Vis. Comput. Graph.*, vol. 13, pp. 1352–1359, 2007.

[16] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal of robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[17] M. A. Lourakis and A. Argyros, "SBA: A Software Package for Generic Sparse Bundle Adjustment," *ACM Trans. Math. Software*, vol. 36, no. 1, pp. 1–30, 2009.

[18] M. Brown, A. Majumder, and R. Yang, "Camera-based calibration techniques for seamless multiprojector displays," *IEEE Trans. Vis. Comput. Graph*, vol. 11, no. 2, pp. 193–206, 2005.

[19] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondence," in *Proc. ECCV*, 1994, pp. 151–158.

[20] J. A. Iglesias Guitián, E. Gobbetti, and F. Marton, "View-dependent exploration of massive volumetric models on large scale light field displays," *The Visual Computer*, vol. 26, no. 6–8, pp. 1037–1047, 2010.